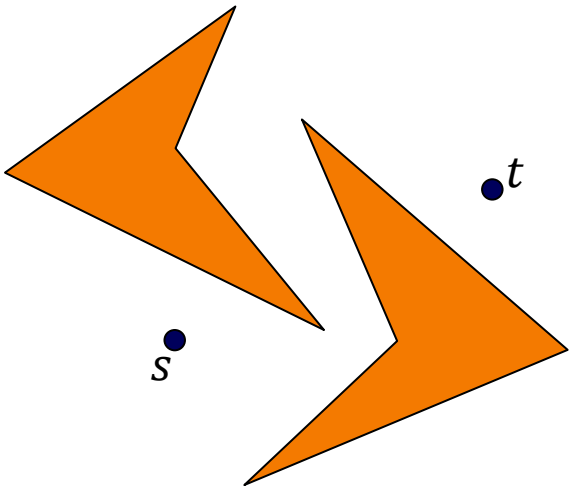
Motion Planning

O’Rourke, Chapter 8

Shortest Path

Goal:

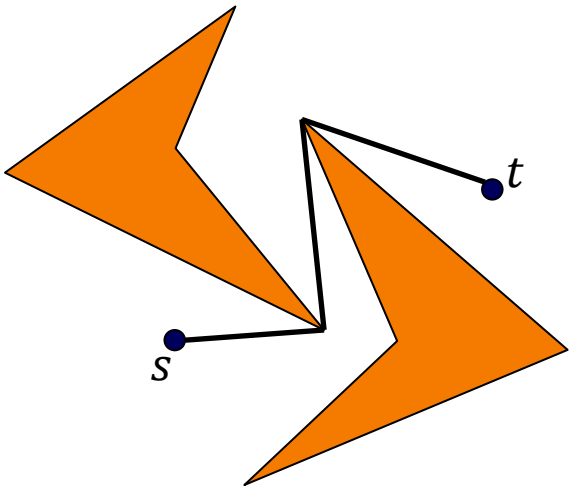
Given a collection of disjoint polygons in the plane, and give a source s and target t, find the **shortest** path from s to t that avoids the polygons.



Shortest Path

Goal:

Given a collection of disjoint polygons in the plane, and give a source s and target t, find the **shortest** path from s to t that avoids the polygons.



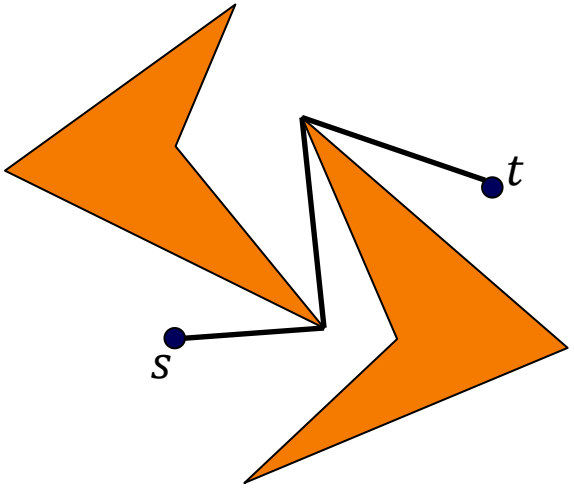
Shortest Path

Goal:

Given a collection of disjoint polygons in the plane, and give a source s and target t, find the **shortest** path from s to t that avoids the polygons.

Key Idea:

Since the path is shortest, it only bends at corners and will otherwise consist of straight segments between vertices and the source/target.



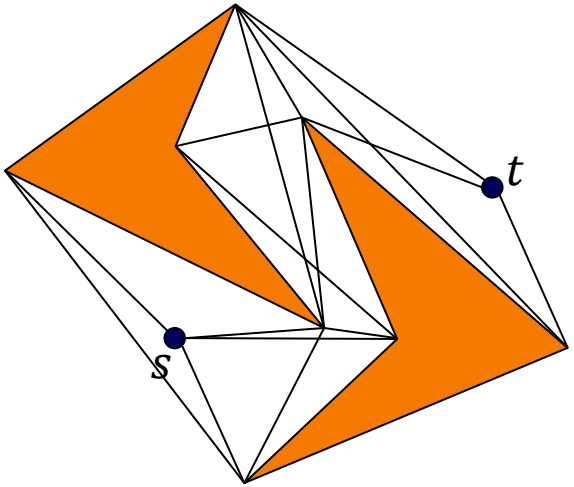
Shortest Path

Goal:

Given a collection of disjoint polygons in the plane, and give a source s and target t, find the **shortest** path from s to t that avoids the polygons.

Approach:

- Construct the *visibility graph*: A graph whose nodes are vertices of the polygons plus s and t, with edges between nodes that "see" each other. (These include polygon edges.)



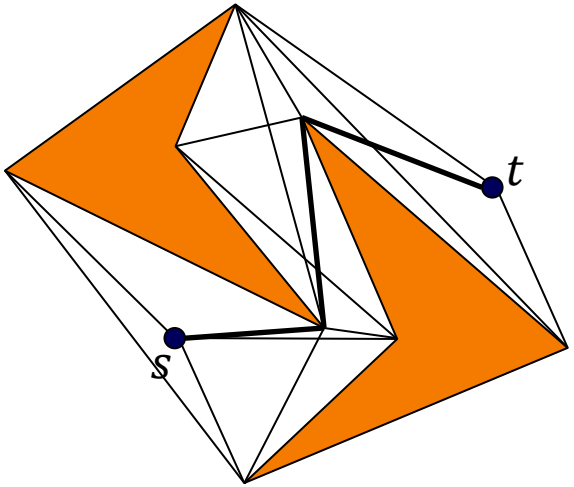
Shortest Path

Goal:

Given a collection of disjoint polygons in the plane, and give a source s and target t, find the **shortest** path from s to t that avoids the polygons.

Approach:

- Construct the *visibility graph*
- Find the shortest path from the source to the target in this graph.

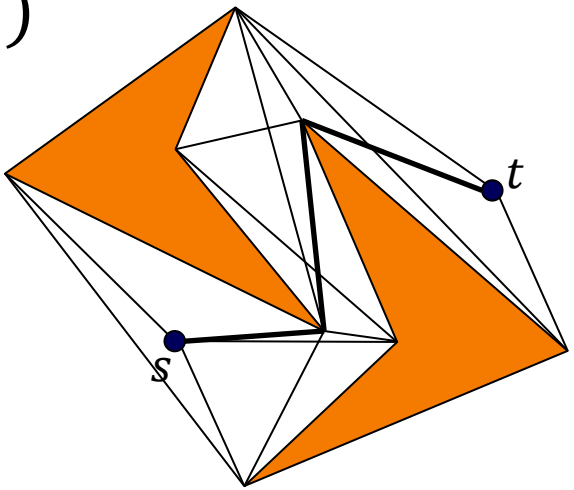


Shortest Path

Find the Shortest Path:

Setting the weight of each edge to the Euclidean distance between its endpoints, this can be solved using Dijkstra's Algorithm.

Complexity: $O(|V|\log|V| + |E|)$



Dijkstra's Shortest Path Algorithm

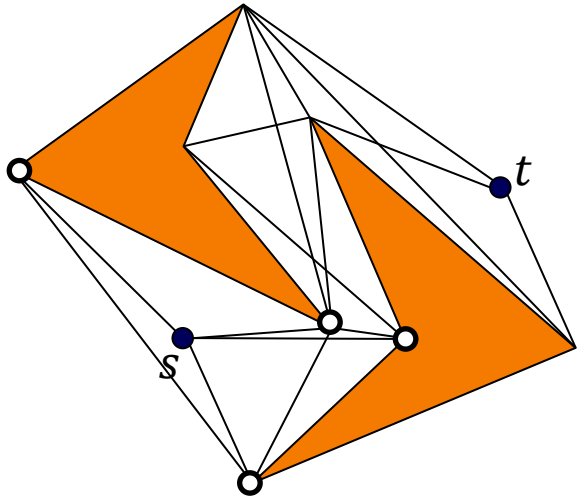
Notation:

- Set V to be the nodes of the graph.
- Set $E \subset V \times V$ to be the set of edges.
- Set $\omega: E \to \mathbb{R}^{\geq 0}$ to be the edge lengths.
- Set $s, t \in V$ to be the source and target vertices.
- Set $d: V \times V \to \mathbb{R} \cup \{\infty\}$ to be the shortest distance function.
  (It could be infinite if the nodes are not connected in the graph.)
- Set $d_s: V \to \mathbb{R} \cup \{\infty\}$ to be the shortest distance from the source.

Dijkstra's Shortest Path Algorithm

Key Idea:

Suppose we computed the distance for the subset of vertices $V_\varepsilon \subset V$ that are within ε of the source.

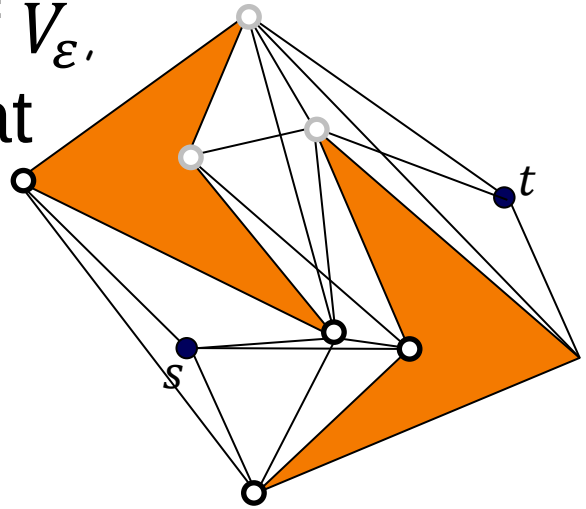


Dijkstra's Shortest Path Algorithm

Key Idea:

Suppose we computed the distance for the subset of vertices $V_\varepsilon \subset V$ that are within ε of the source.

Let $N(V_\varepsilon)$ be the *neighbors* of V_ε, i.e. all vertices $v \in V$ such that $(u, v) \in E$ for some $u \in V_\varepsilon$.

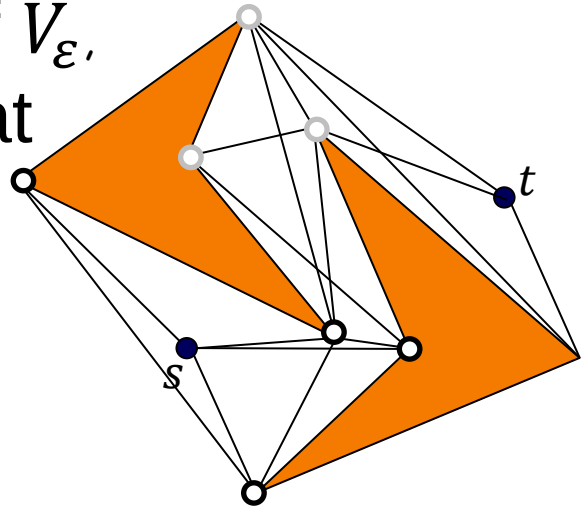


Dijkstra's Shortest Path Algorithm

Key Idea:

Suppose we computed the distance for the subset of vertices $V_\varepsilon \subset V$ that are within ε of the source.

Let $N(V_\varepsilon)$ be the *neighbors* of V_ε, i.e. all vertices $v \in V$ such that $(u, v) \in E$ for some $u \in V_\varepsilon$.



Let $d_s^\varepsilon : N(V_\varepsilon) \to \mathbb{R}$ be the candidate distance from s:

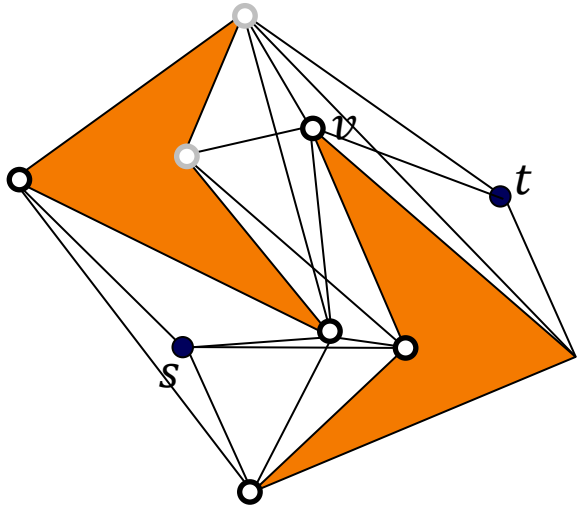
$$d_s^\varepsilon(v) = \min_{(u,v) \in V_\varepsilon \times N(V_\varepsilon) \cap E} d_s(u) + \omega(u, v)$$

Dijkstra's Shortest Path Algorithm

Key Idea:

The vertex $v \in N(V_\varepsilon)$ / V_ε minimizing d_s^ε satisfies:

$$d_s(v) = d_s^\varepsilon(v)$$



Dijkstra's Shortest Path Algorithm

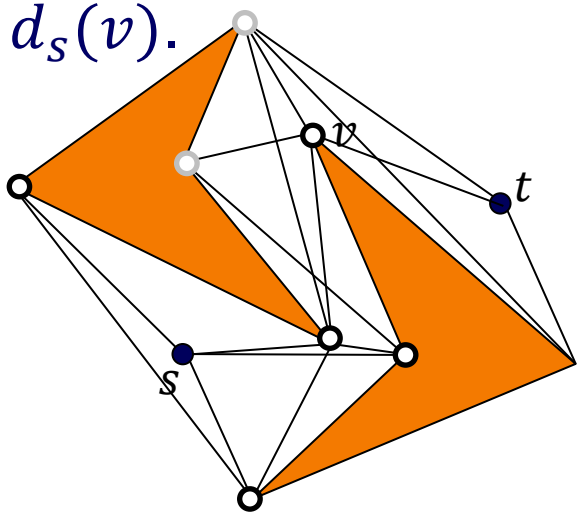
Key Idea:

The vertex $v \in N(V_\varepsilon)$ / V_ε minimizing d_s^ε satisfies:

$$d_s(v) = d_s^\varepsilon(v)$$

Proof:

Assume to the contrary $d_s^\varepsilon(v) > d_s(v)$.



Dijkstra's Shortest Path Algorithm

Key Idea:

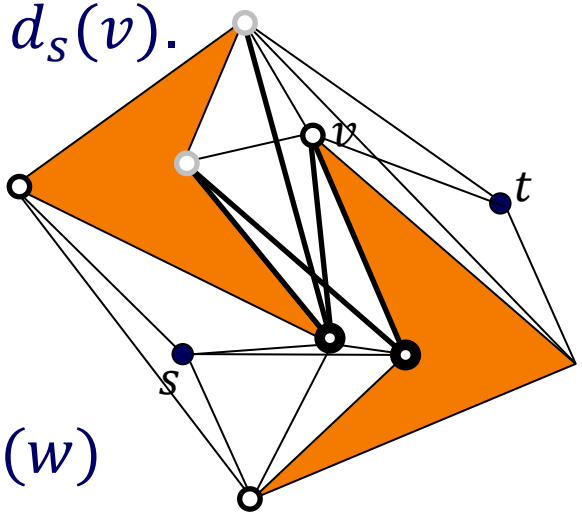
The vertex $v \in N(V_\varepsilon)$ / V_ε minimizing d_s^ε satisfies:

$$d_s(v) = d_s^\varepsilon(v)$$

Proof:

Assume to the contrary $d_s^\varepsilon(v) > d_s(v)$.

The shortest path to v must go through some $u \in V_\varepsilon$ and use an edge $e = (u, w)$ with $w \notin V_\varepsilon$.



$\Rightarrow d_s(v) \geq d_s(u) + \omega(u, w)$

$\Rightarrow d_s^\varepsilon(v) > d_s(u) + \omega(u, w) \geq d_s^\varepsilon(w)$

$\Rightarrow$ The vertex v does not minimize d_s^ε.

Dijkstra's Shortest Path Algorithm

Implementation:

Maintain a set of known distances, $d_s: V \to \mathbb{R} \cup \{\infty\}$, from the source to vertices in V. (Vertices with finite distances define the set V_ε).

Maintain a map, $p: V \to V \cup \{\emptyset\}$, giving the previous vertex on the path from the source. (Or no vertex if we are either at the source or haven't computed the path yet.)

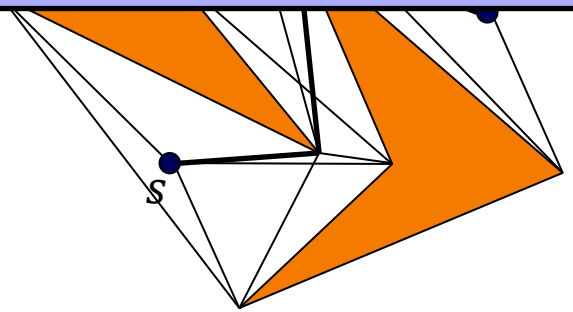
Maintain a queue, $Q \subset \mathbb{R} \times V \times V$, of candidate distances/vertices/previous for the current neighbors. (Vertices in the queue define $N(V_\varepsilon)$.)

Dijkstra's Shortest Path Algorithm

Dijkstra($G = (V, E, \omega: E \to \mathbb{R})$):

- »$\forall\, v \in V$: $d_s[v] \leftarrow \infty$
- »$\forall\, v \in V$: $p[v] \leftarrow \emptyset$
- »$Q \leftarrow \{(0, s, \emptyset)\}$

Initialize the shortest-path distances to *unknown*.
Initialize the previous-on-the-path vertices to *unknown*.
Initialize the queue to contain the source.

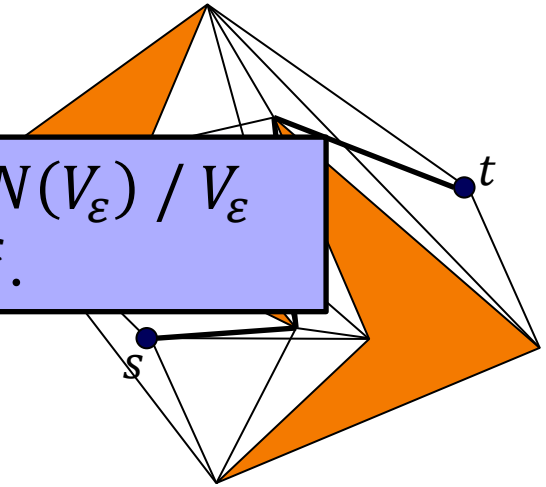


Dijkstra's Shortest Path Algorithm

Dijkstra($G = (V, E, \omega: E \to \mathbb{R})$):

- »$\forall\, v \in V$: $d_s[v] \leftarrow \infty$
- »$\forall\, v \in V$: $p[v] \leftarrow \emptyset$
- »$Q \leftarrow \{(0, s, \emptyset)\}$
- »while not_empty(Q)
  - $(v, d, \pi) \leftarrow$ pop_smallest(Q)
  - if($d < d_s[v]$)

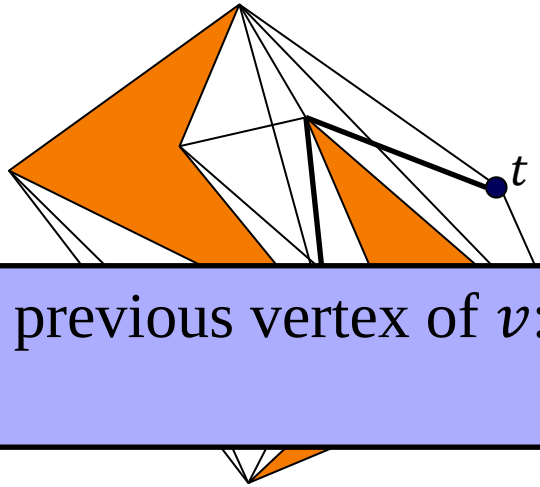
Find the neighbor $v \in N(V_\varepsilon)$ / V_ε minimizing d_s^ε.



Dijkstra's Shortest Path Algorithm

Dijkstra($G = (V, E, \omega: E \to \mathbb{R})$):

- »$\forall\, v \in V$: $d_s[v] \leftarrow \infty$
- »$\forall\, v \in V$: $p[v] \leftarrow \emptyset$
- »$Q \leftarrow \{(0, s, \emptyset)\}$
- »while not_empty(Q)
  - $(v, d, \pi) \leftarrow$ pop_smallest(Q)
  - if($d < d_s[v]$)
    - $d_s[v] \leftarrow d$
    - $p[v] \leftarrow \pi$



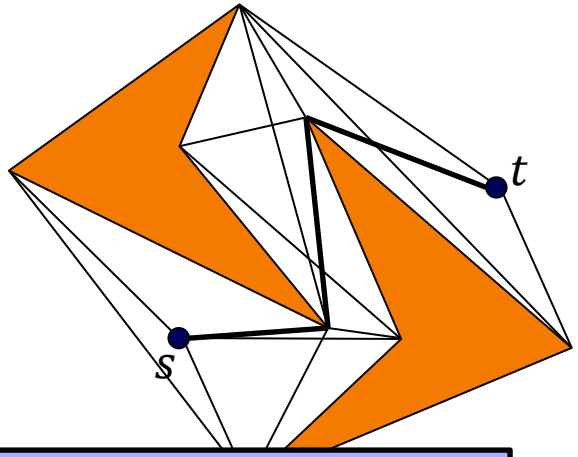
Update the shortest distance and previous vertex of v:

$$V_\varepsilon = V_{d_s(v)}$$

Dijkstra's Shortest Path Algorithm

Dijkstra($G = (V, E, \omega: E \to \mathbb{R})$):

- »$\forall\, v \in V$: $d_s[v] \leftarrow \infty$
- »$\forall\, v \in V$: $p[v] \leftarrow \emptyset$
- »$Q \leftarrow \{(0, s, \emptyset)\}$
- »while not_empty(Q)
  - $(v, d, \pi) \leftarrow$ pop_smallest(Q)
  - if($d < d_s[v]$)
    - $d_s[v] \leftarrow d$
    - $p[v] \leftarrow \pi$
    - $\forall (v, w) \in E$
      - $d' \leftarrow d + \omega(v, w)$
      - $Q \leftarrow Q \cup \{(d', w, v)\}$

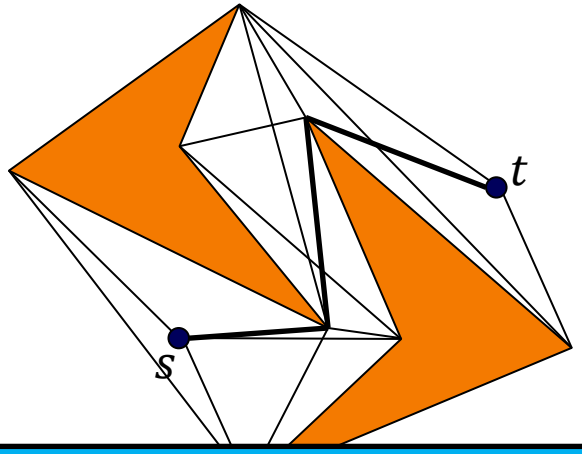


Update the queue to include neighbors of v:
$N(V_\varepsilon) = N(V_{d_s(v)})$

Dijkstra's Shortest Path Algorithm

Dijkstra($G = (V, E, \omega: E \to \mathbb{R})$):

- »$\forall\, v \in V$: $d_s[v] \leftarrow \infty$
- »$\forall\, v \in V$: $p[v] \leftarrow \emptyset$
- »$Q \leftarrow \{(0, s, \emptyset)\}$
- »while not_empty(Q)
  - $(v, d, \pi) \leftarrow$ pop_smallest(Q)
  - if($d < d_s[v]$)
    - $d_s[v] \leftarrow d$
    - $p[v] \leftarrow \pi$
    - $\forall (v, w) \in E$
      - $d' \leftarrow d + \omega(v, w)$
      - $Q \leftarrow Q \cup \{(d', w, v)\}$



If($d_s[t] = \infty$): t is unreachable

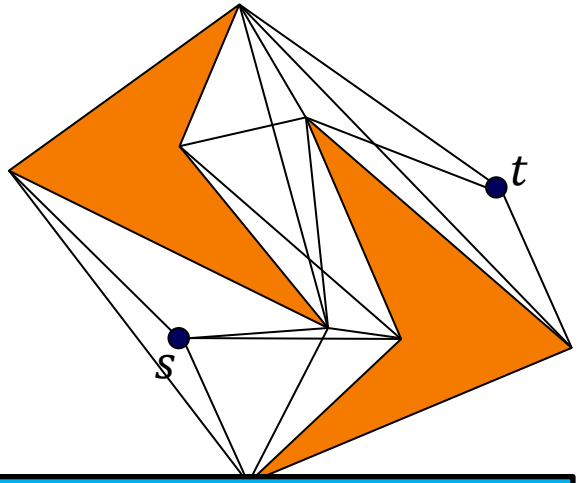
Else: the path from t back to s is $\{t, p[t], p[p[t]], \ldots, s\}$

Visibility Graph

Naïve Algorithm:

For each pair of nodes and each polygon edge test if the segment between the nodes intersects the edge.

Complexity: $O(|V|^3)$

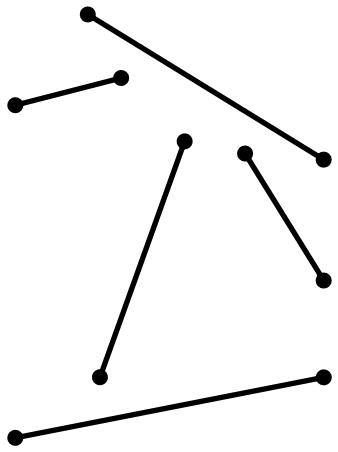


Note that the visibility graph can have $O(|V|^2)$ edges.

Visibility Graph [Lee 1987]

Given non-intersecting edges $E \subset V \times V$, construct the visibility graph by first constructing the reachability function:

$$R: V \times [-\infty, \infty) \rightarrow E$$

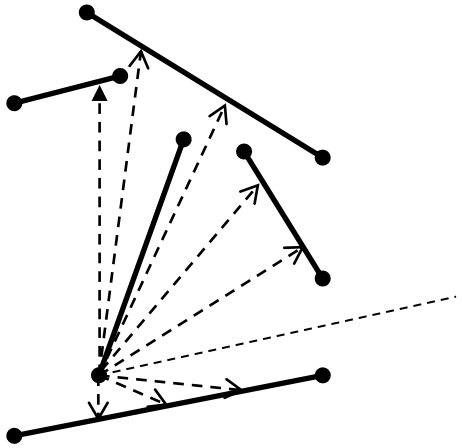


Visibility Graph [Lee 1987]

Given non-intersecting edges $E \subset V \times V$, construct the visibility graph by first constructing the reachability function:

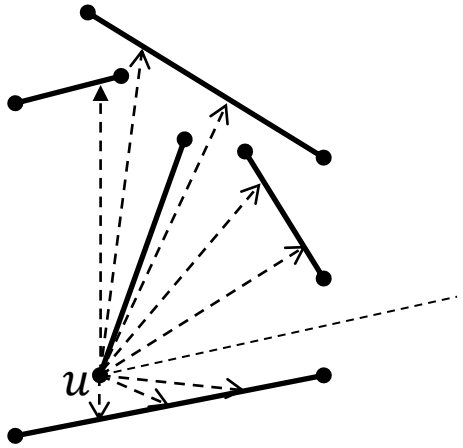
$$R: V \times [-\infty, \infty) \rightarrow E$$

This gives the first edge hit when travelling from $v \in V$ along a line to the right with slope $m \in [-\infty, \infty)$.



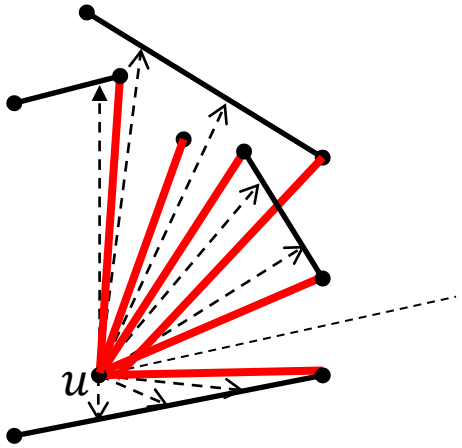
Visibility Graph [Lee 1987]

Because the edges in E are non-intersecting, the label $R(u,\cdot)$ can only change when the slope aligns with a segment $(u,\cdot) \in V \times V$.



Visibility Graph [Lee 1987]

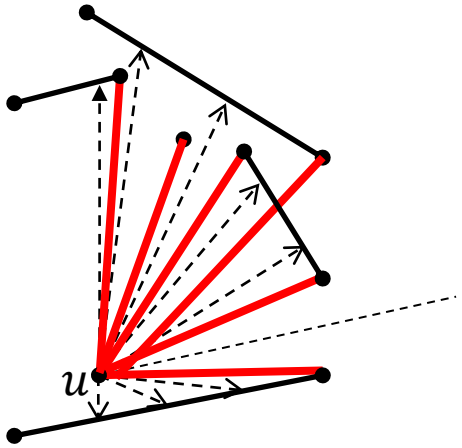
Because the edges in E are non-intersecting, the label $R(u,\cdot)$ can only change when the slope aligns with a segment $(u,\cdot) \in V \times V$.



Visibility Graph [Lee 1987]

Because the edges in E are non-intersecting, the label $R(u,\cdot)$ can only change when the slope aligns with a segment $(u,\cdot) \in V \times V$.

$\Rightarrow$ Instead of considering the continuum $[-\infty, \infty)$, it suffices to consider the $O(|V|^2)$ slopes generated by the vertex pairs $V \times V$.



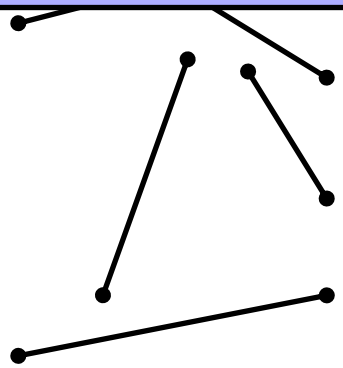
Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

- $S \leftarrow$ SortBySlope($V \times V$)
- Compute $R(\cdot, -\infty)$

Preprocessing:

- Sort the segments by their slopes
- Initialize reachability in the downward direction

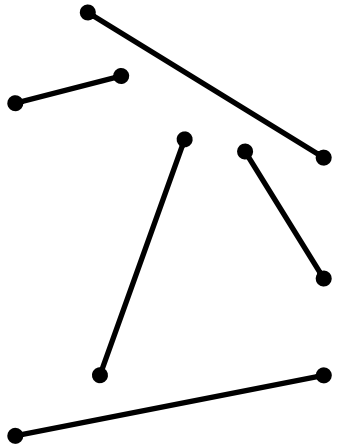


Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

Iterate over the segments:

- If the segment is an edge, its starting point must see its ending point.
  - For $(u, v) \in S$ (w/ $u_x < v_x$)
    - »if($(u, v) \in E$): Output uv



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

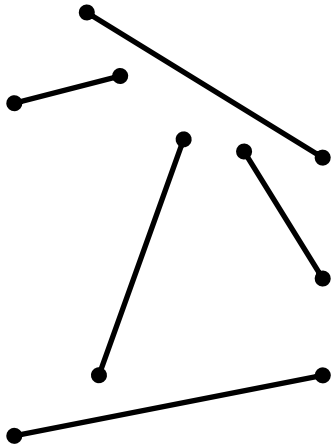
Iterate over the segments:

- Otherwise, compute the intersection of the ray through the segment with the edge previously reached by the segment's starting vertex.

»if($(u, v) \in E$): Output uv

»else:

– $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$



Visibility Graph [Lee 1987]

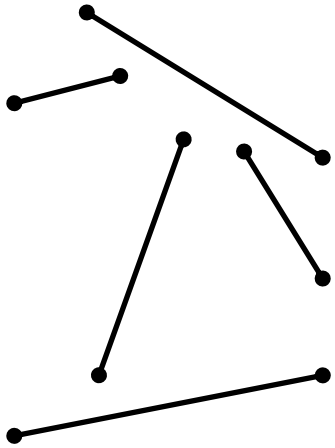
VisbilityGraph($G = (V, E)$):

Iterate over the segments:

- If the intersection is the end of the segment:
  - The start can see the end.
  - The reachability is unchanged.

»else:

- $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
- if($v = w$)
  - Output uv
  - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



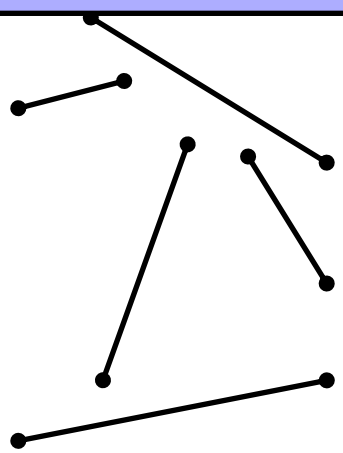
Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

Iterate over the segments:

- If the reachable edge is behind the line segment or the end is closer than the point of intersection:
  - The start can see the end.
  - We can now reach the edge coming out of the end.

- $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
- if($v = w$)
  - Output uv
  - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
- else if($w = \emptyset$ or $|uv| < |uw|$)
  - Output uv
  - $R(u, \angle uv) \leftarrow e(v)$



Visibility Graph [Lee 1987]

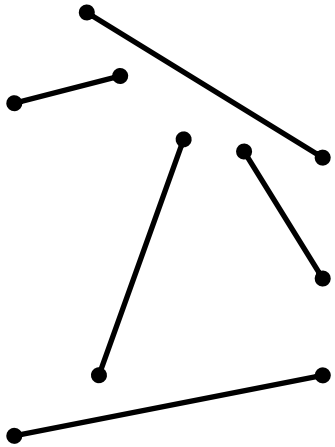
VisbilityGraph($G = (V, E)$):

Iterate over the segments:

- Otherwise:
  - The start cannot see the end (it's blocked by the reachable edge).
  - The reachability is unchanged.

»else:

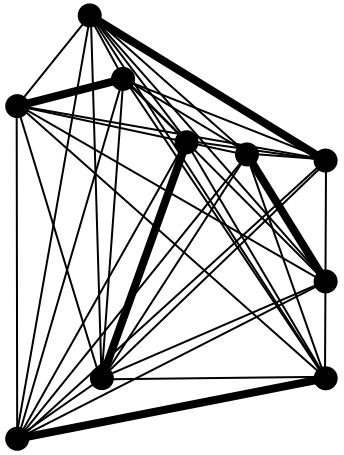
- $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
- if($v = w$)
  - Output uv
  - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
- else if($w = \emptyset$ or $|uv| < |uw|$)
  - Output uv
  - $R(u, \angle uv) \leftarrow e(v)$
- else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

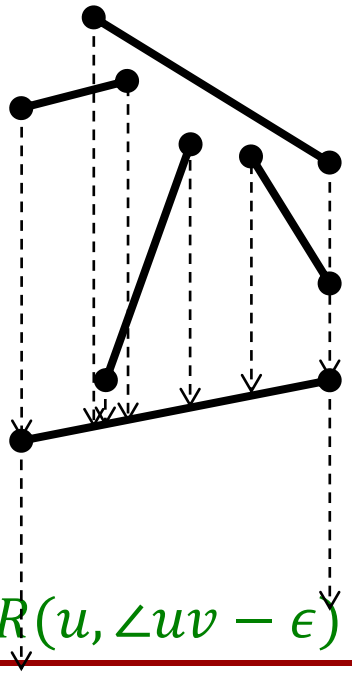
- $S \leftarrow$SortBySlope($V \times V$)
- Compute $R(\cdot, -\infty)$
- For $(u, v) \in S$ (w/ $u_x < v_x$)
  - »if($(u, v) \in E$): Output uv
  - »else:
    - $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
    - if($v = w$)
      - Output uv
      - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
    - else if($w = \emptyset$ or $|uv| < |uw|$)
      - Output uv
      - $R(u, \angle uv) \leftarrow e(v)$
    - else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

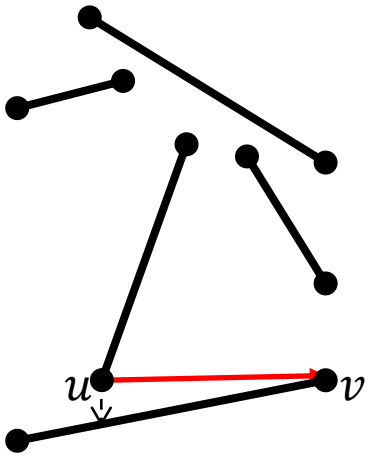
- $S \leftarrow$ SortBySlope($V \times V$)
- Compute $R(\cdot, -\infty)$
- For $(u, v) \in S$ (w/ $u_x < v_x$)
  - »if($(u, v) \in E$): Output uv
  - »else:
    - $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
    - if($v = w$)
      - Output uv
      - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
    - else if($w = \emptyset$ or $|uv| < |uw|$)
      - Output uv
      - $R(u, \angle uv) \leftarrow e(v)$
    - else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

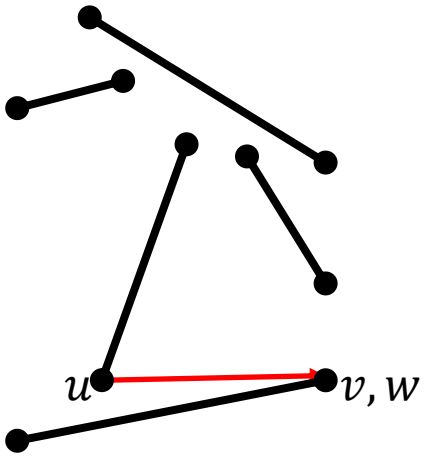
- $S \leftarrow$ SortBySlope($V \times V$)
- Compute $R(\cdot, -\infty)$
- For $(u, v) \in S$ (w/ $u_x < v_x$)
  - »if($(u, v) \in E$): Output uv
  - »else:
    - $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
    - if($v = w$)
      - Output uv
      - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
    - else if($w = \emptyset$ or $|uv| < |uw|$)
      - Output uv
      - $R(u, \angle uv) \leftarrow e(v)$
    - else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

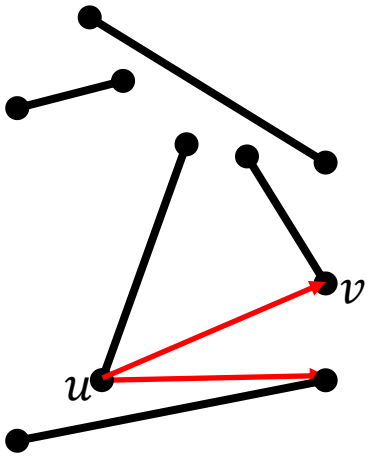
- $S \leftarrow$ SortBySlope($V \times V$)
- Compute $R(\cdot, -\infty)$
- For $(u, v) \in S$ (w/ $u_x < v_x$)
  - »if($(u, v) \in E$): Output uv
  - »else:
    - $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
    - if($v = w$)
      - Output uv
      - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
    - else if($w = \emptyset$ or $|uv| < |uw|$)
      - Output uv
      - $R(u, \angle uv) \leftarrow e(v)$
    - else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

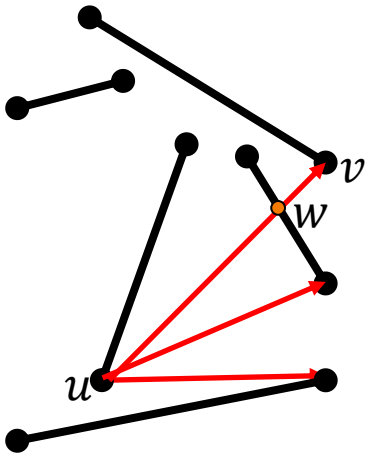
- $S \leftarrow$ SortBySlope($V \times V$)
- Compute $R(\cdot, -\infty)$
- For $(u, v) \in S$ (w/ $u_x < v_x$)
  - »if($(u, v) \in E$): Output uv
  - »else:
    - $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
    - if($v = w$)
      - Output uv
      - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
    - else if($w = \emptyset$ or $|uv| < |uw|$)
      - Output uv
      - $R(u, \angle uv) \leftarrow e(v)$
    - else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

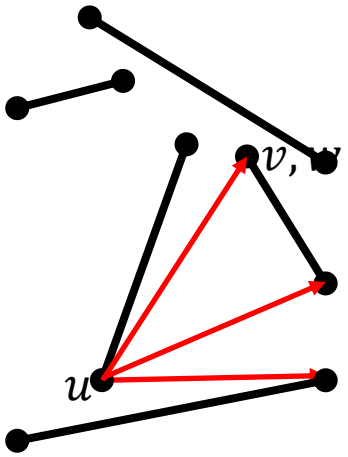
- $S \leftarrow$ SortBySlope($V \times V$)
- Compute $R(\cdot, -\infty)$
- For $(u, v) \in S$ (w/ $u_x < v_x$)
  - »if($(u, v) \in E$): Output uv
  - »else:
    - $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
    - if($v = w$)
      - Output uv
      - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
    - else if($w = \emptyset$ or $|uv| < |uw|$)
      - Output uv
      - $R(u, \angle uv) \leftarrow e(v)$
    - else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

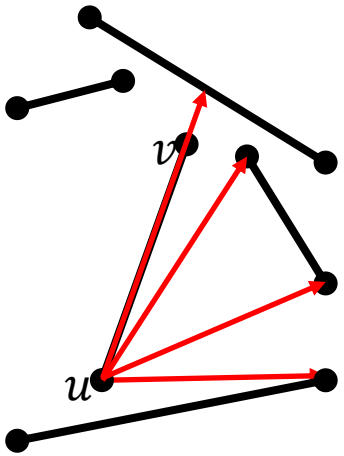
- $S \leftarrow$ SortBySlope($V \times V$)
- Compute $R(\cdot, -\infty)$
- For $(u, v) \in S$ (w/ $u_x < v_x$)
  - »if($(u, v) \in E$): Output uv
  - »else:
    - $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
    - if($v = w$)
      - Output uv
      - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
    - else if($w = \emptyset$ or $|uv| < |uw|$)
      - Output uv
      - $R(u, \angle uv) \leftarrow e(v)$
    - else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

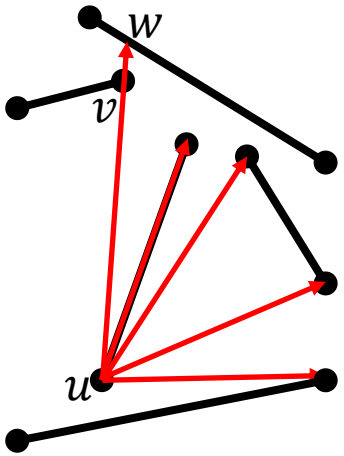
- $S \leftarrow$ SortBySlope($V \times V$)
- Compute $R(\cdot, -\infty)$
- For $(u, v) \in S$ (w/ $u_x < v_x$)
  - »if($(u, v) \in E$): Output uv
  - »else:
    - $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
    - if($v = w$)
      - Output uv
      - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
    - else if($w = \emptyset$ or $|uv| < |uw|$)
      - Output uv
      - $R(u, \angle uv) \leftarrow e(v)$
    - else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

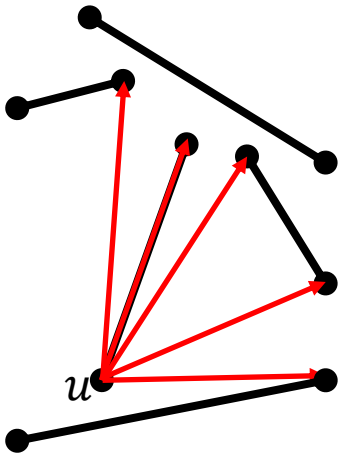
- $S \leftarrow$ SortBySlope($V \times V$)
- Compute $R(\cdot, -\infty)$
- For $(u, v) \in S$ (w/ $u_x < v_x$)
  - »if($(u, v) \in E$): Output uv
  - »else:
    - $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
    - if($v = w$)
      - Output uv
      - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
    - else if($w = \emptyset$ or $|uv| < |uw|$)
      - Output uv
      - $R(u, \angle uv) \leftarrow e(v)$
    - else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

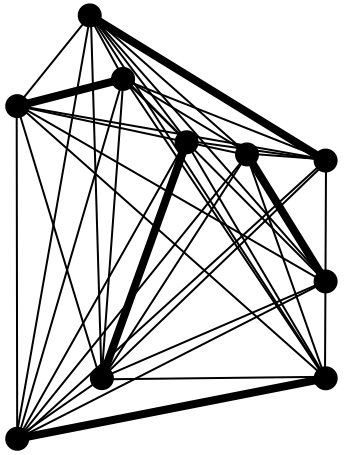
- $S \leftarrow$ SortBySlope($V \times V$)
- Compute $R(\cdot, -\infty)$
- For $(u, v) \in S$ (w/ $u_x < v_x$)
  - »if($(u, v) \in E$): Output uv
  - »else:
    - $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
    - if($v = w$)
      - Output uv
      - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
    - else if($w = \emptyset$ or $|uv| < |uw|$)
      - Output uv
      - $R(u, \angle uv) \leftarrow e(v)$
    - else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

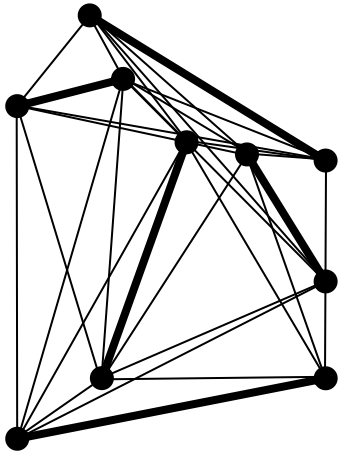
- $S \leftarrow$ SortBySlope($V \times V$)
- Compute $R(\cdot, -\infty)$
- For $(u, v) \in S$ (w/ $u_x < v_x$)
  - »if($(u, v) \in E$): Output uv
  - »else:
    - $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
    - if($v = w$)
      - Output uv
      - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
    - else if($w = \emptyset$ or $|uv| < |uw|$)
      - Output uv
      - $R(u, \angle uv) \leftarrow e(v)$
    - else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

- $S \leftarrow$ SortBySlope($V \times V$)
- Compute $R(\cdot, -\infty)$
- For $(u, v) \in S$ (w/ $u_x < v_x$)
  - »if($(u, v) \in E$): Output uv
  - »else:
    - $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
    - if($v = w$)
      - Output uv
      - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
    - else if($w = \emptyset$ or $|uv| < |uw|$)
      - Output uv
      - $R(u, \angle uv) \leftarrow e(v)$
    - else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

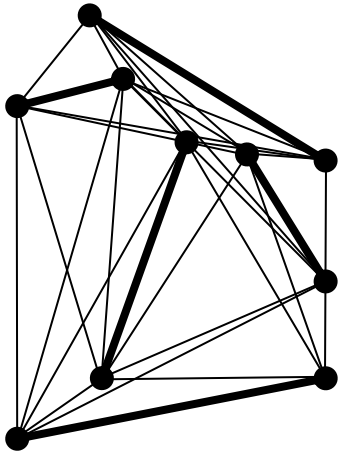
- $S \leftarrow$ Sor
- Comput
- For $(u,$

Because segments are sorted, when we process slope $\angle uv$, we can assume that $R(\cdot, \angle uv - \epsilon)$ has been set correctly

»if($(u, v) \in E$): Output uv

»else:

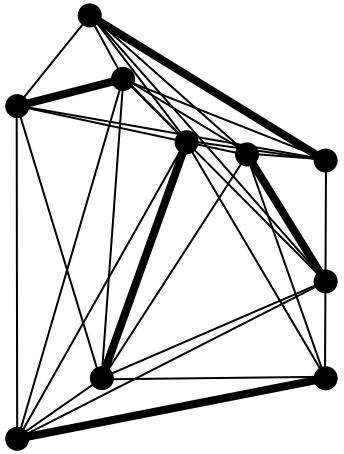
- $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
- if($v = w$)
  - Output uv
  - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
- else if($w = \emptyset$ or $|uv| < |uw|$)
  - Output uv
  - $R(u, \angle uv) \leftarrow e(v)$
- else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

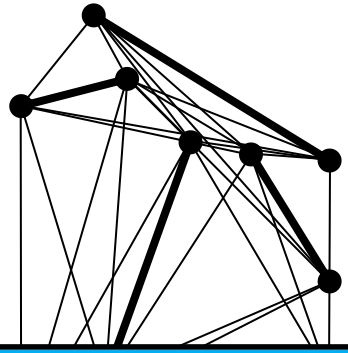
- $S \leftarrow$ SortBySlope($V \times V$) ← $O(|V|^2 \log |V|)$
- Compute $R(\cdot, -\infty)$ ← $O(|V|^2)$
- For $(u, v) \in S$ (w/ $u_x < v_x$) — $O(|V|^2)$
  - »if($(u, v) \in E$): Output uv
  - »else:
    - $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
    - if($v = w$)
      - Output uv
      - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
    - else if($w = \emptyset$ or $|uv| < |uw|$)
      - Output uv
      - $R(u, \angle uv) \leftarrow e(v)$
    - else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



Visibility Graph [Lee 1987]

VisbilityGraph($G = (V, E)$):

- $S \leftarrow$ SortBySlope($V \times V$) ← $O(|V|^2 \log |V|)$
- Compute $R(\cdot, -\infty)$ ← $O(|V|^2)$
- For $(u, v) \in S$ (w/ $u_x < v_x$)
  - »if($(u, v) \in E$): Output uv
  - »else:
    - $w \leftarrow R(u, \angle uv - \epsilon) \cap \overrightarrow{uv}$
    - if($v = w$)
      - Output uv
      - $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$
    - Output u
    - $R(u, \angle uv) \leftarrow e(v)$
    - else if($|uv| > |uw|$): $R(u, \angle uv) \leftarrow R(u, \angle uv - \epsilon)$



$O(|V|^2)$

The $O(|V|^2 \log |V|)$ complexity comes from sorting.

Can we sort more quickly?

Visibility Graph [Welzl 1985]

To compute visibility graph, we didn't require that edges in $S = V \times V$ be fully sorted.

Rather:

- For all $u \in V$ and any $v_1, v_2 \in V$, we require the segments to be sorted so that we encounter segment (u, v_1) before (u, v_2) whenever the slope of $\overline{uv_1}$ is less than the slope of $\overline{uv_2}$.

Visibility Graph [Welzl 1985]

Recall:

- An arrangement of N lines can be computed in $O(N^2)$ time.

Visibility Graph [Welzl 1985]

Recall:

- An arrangement of N lines can be computed in $O(N^2)$ time.
- Given a point $p = (\alpha, \beta) \in \mathbb{R}^2$, it's dual is the line:

$$D(p) = \{(x, y) | y = 2\alpha x - \beta\}$$

Visibility Graph [Welzl 1985]

Recall:

- An arrangement of N lines can be computed in $O(N^2)$ time.
- Given a point $p = (\alpha, \beta) \in \mathbb{R}^2$, it's dual is the line:

$$D(p) = \{(x, y) | y = 2\alpha x - \beta\}$$

- Given a line $L = \{(x, y) | y = mx + b\} \subset \mathbb{R}^2$, it's dual is the point:

$$D(L) = \left(\frac{m}{2}, -b\right)$$

Visibility Graph [Welzl 1985]

$$D(p) = \{(x, y) | y = 2\alpha x - \beta\}$$
$$D(L) = \left(\frac{m}{2}, -b\right)$$

In Particular:

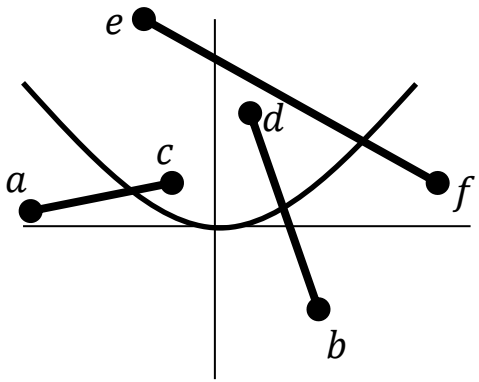
Given a line $L = \{(x, y) | y = mx + b\}$:

- The duals $D(p)$ of points $p \in L$ will be lines intersecting through $D(L)$.
- The slope of the dual line $D(x, mx + b)$ will increase as x increases.

Visibility Graph [Welzl 1985]

Approach:

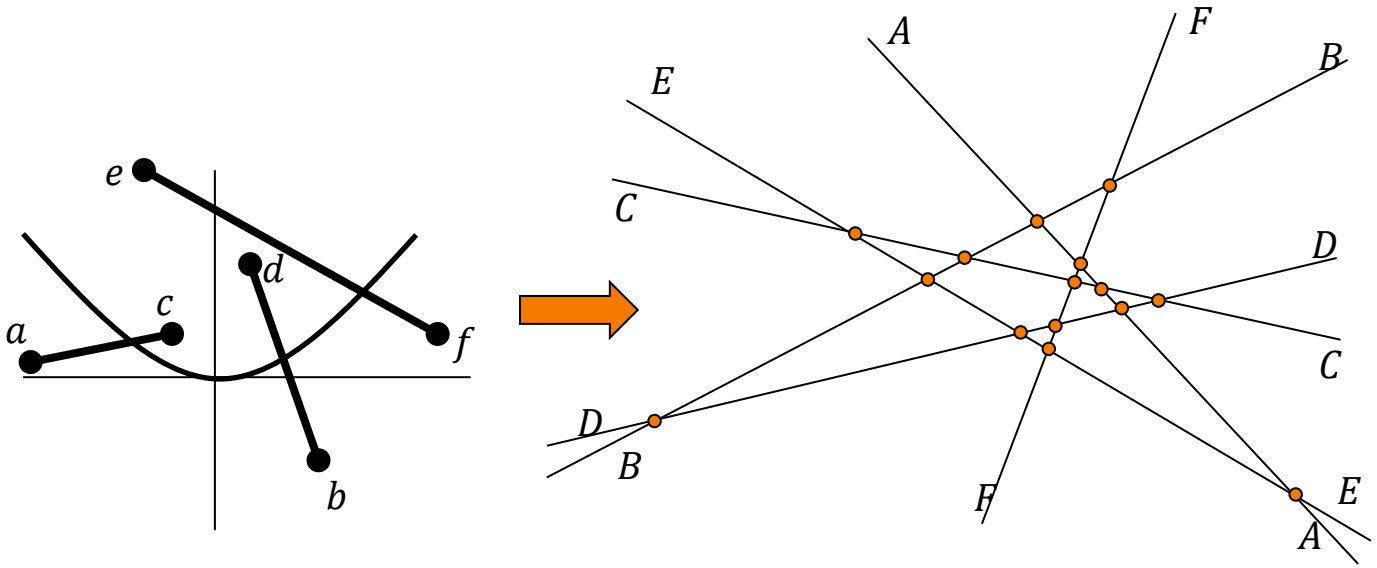
- Compute the line arrangement dual to V.



Visibility Graph [Welzl 1985]

Approach:

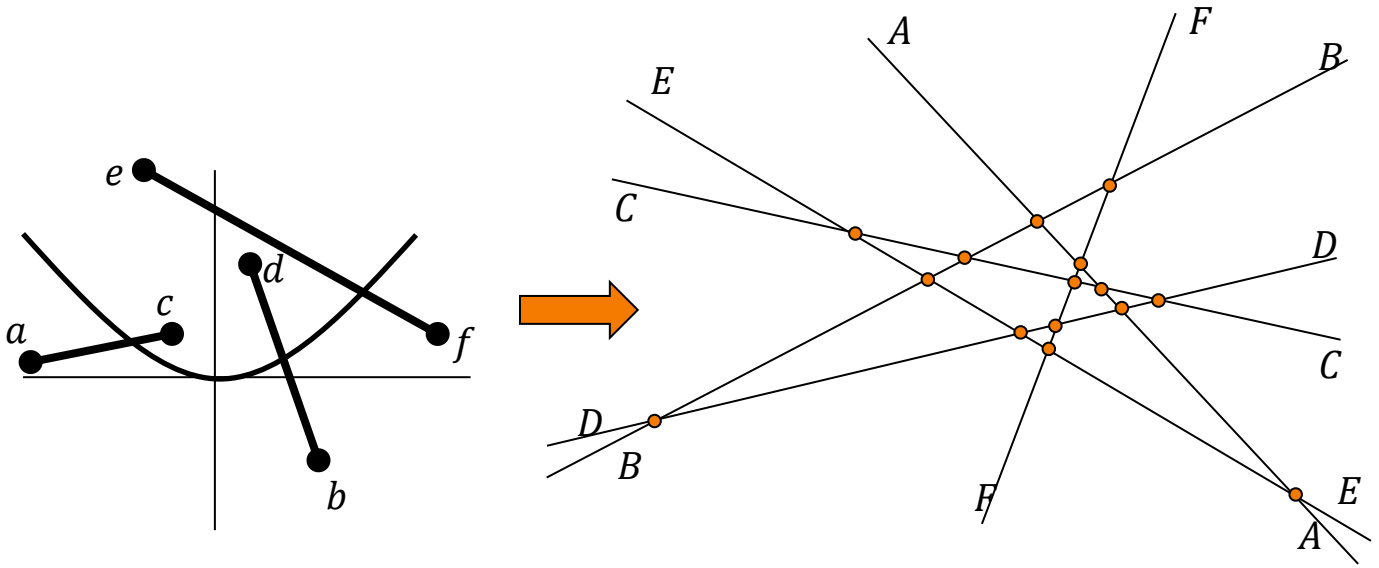
- Compute the line arrangement dual to V.



Visibility Graph [Welzl 1985]

Approach:

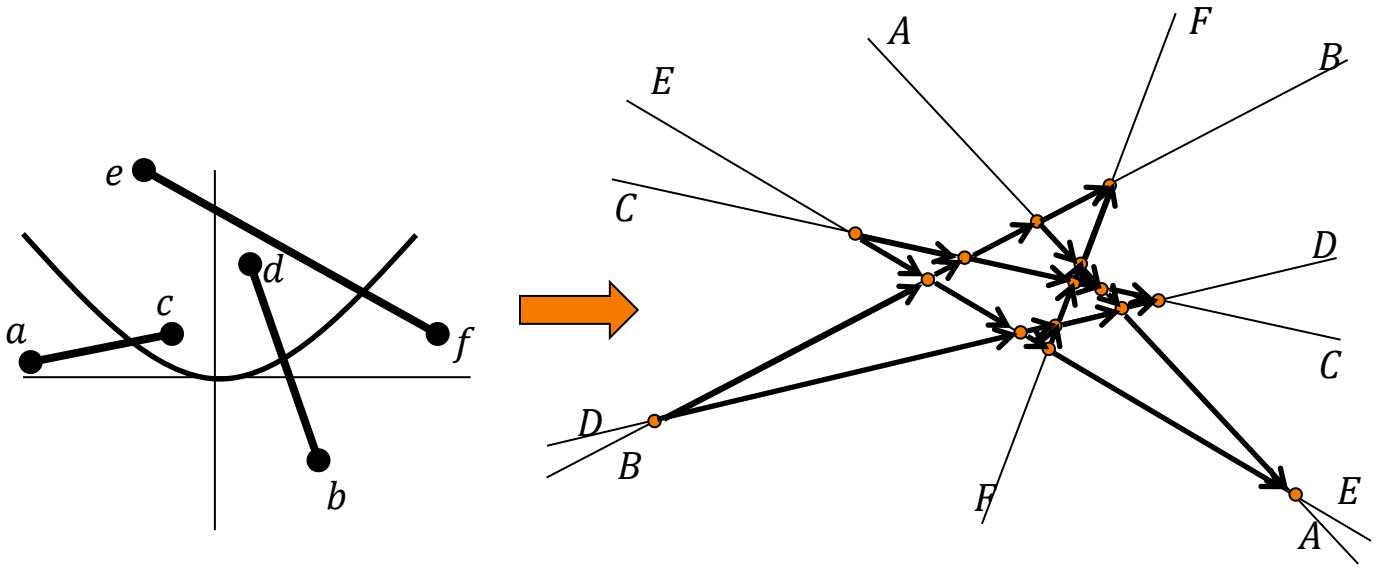
- Compute the line arrangement dual to V.
- Define a directed graph on the dual:
  - $v \to w$ if v and w are adjacent and $v_x < w_x$.



Visibility Graph [Welzl 1985]

Approach:

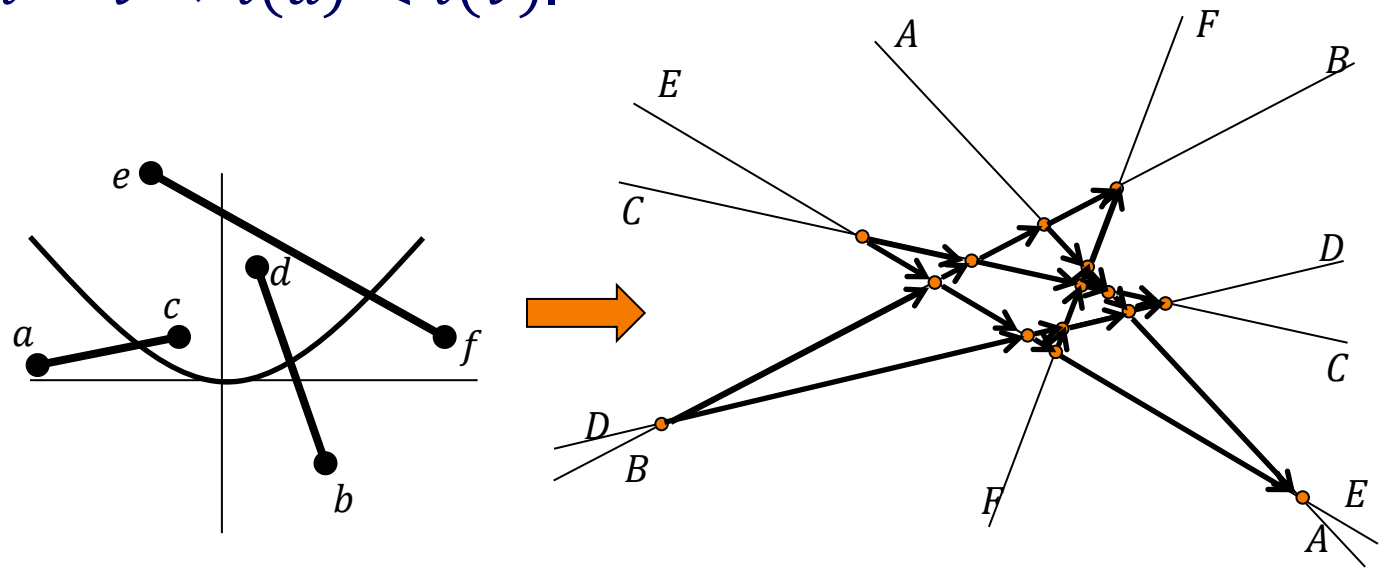
- Compute the line arrangement dual to V.
- Define a directed graph on the dual:
  - $v \to w$ if v and w are adjacent and $v_x < w_x$.



Visibility Graph [Welzl 1985]

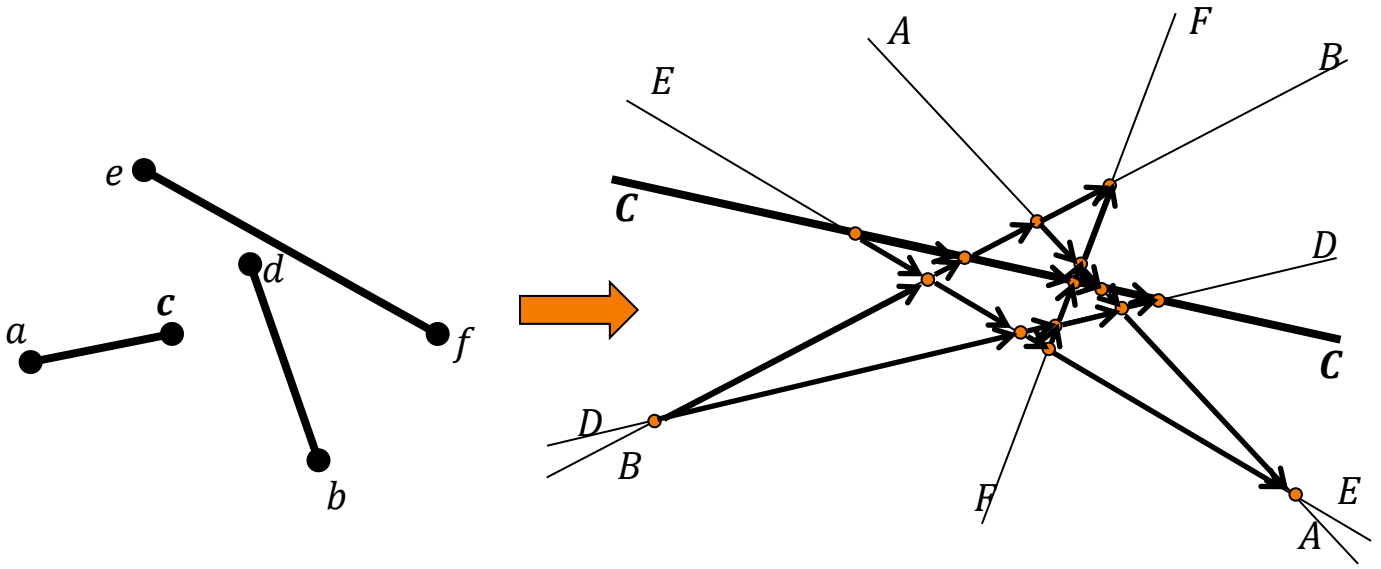
Approach:

- Compute the line arrangement dual to V.
- Define a directed graph on the dual.
- Perform a topological sort on the directed graph:
  - Assign integer indices to the nodes such that $u \to v \Rightarrow i(u) < i(v)$.



Visibility Graph [Welzl 1985]

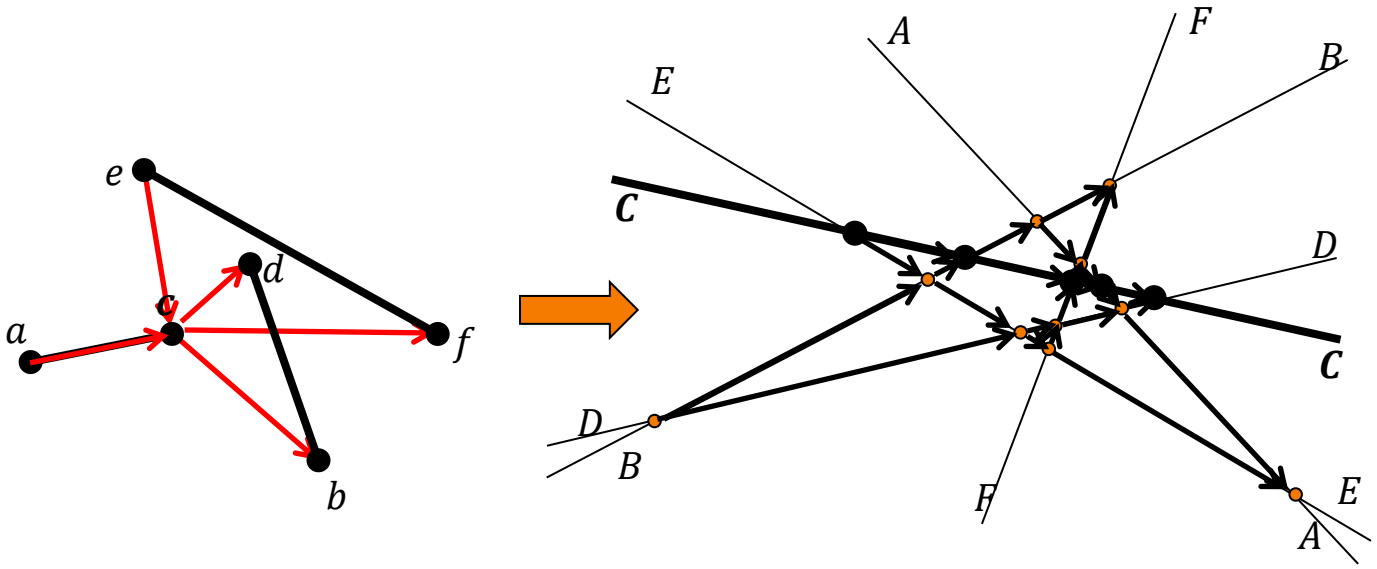
Consider a dual line, $C = D(c)$, in the arrangement.



Visibility Graph [Welzl 1985]

Consider a dual line, $C = D(c)$, in the arrangement.

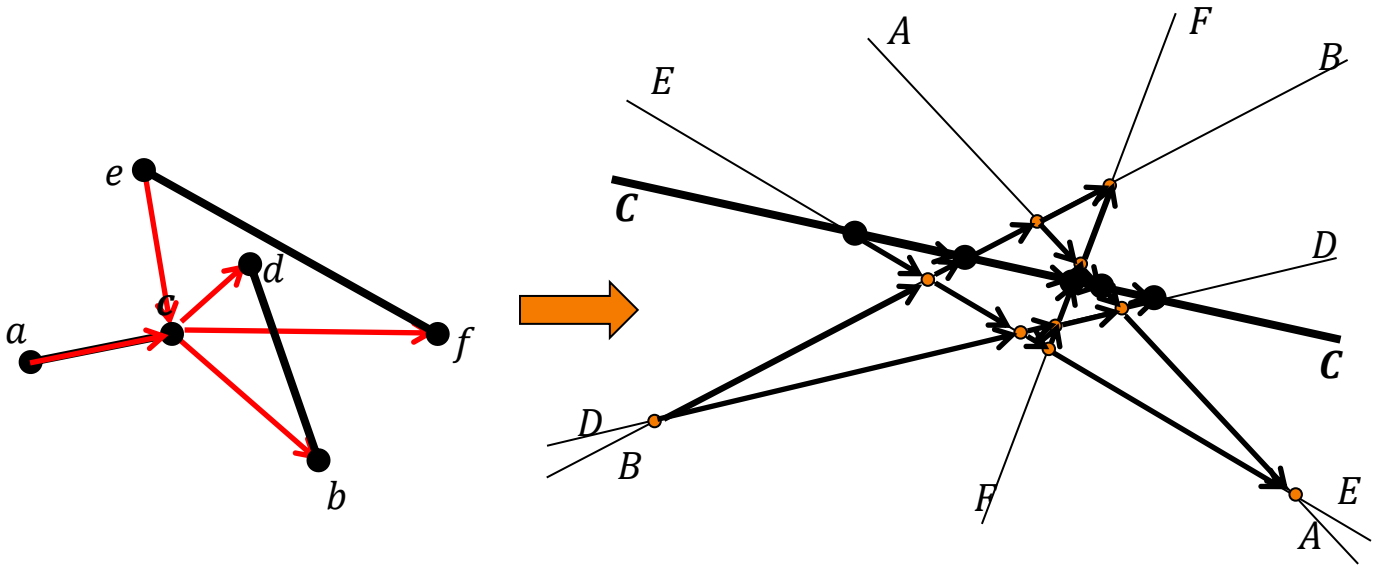
- Intersections on C correspond to primal segments having c as an endpoint.



Visibility Graph [Welzl 1985]

Consider a dual line, $C = D(c)$, in the arrangement.

- Intersections on C correspond to primal segments having c as an endpoint.
- If the indices are sorted topologically, indices increase from left to right along C.

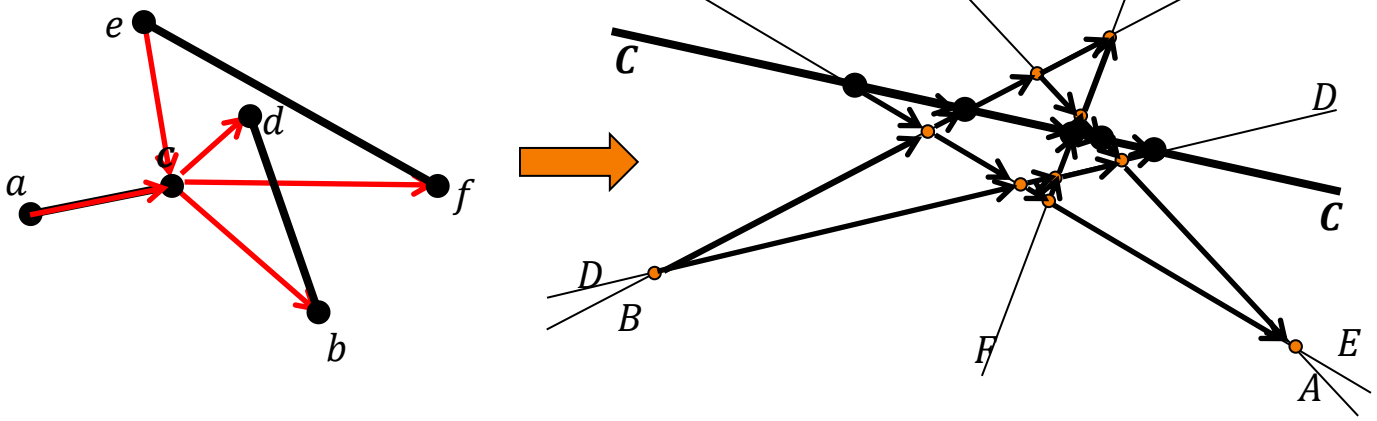


Visibility Graph [Welzl 1985]

Consider a dual line, $C = D(c)$, in the arrangement.

- Intersections on C correspond to primal segments having c as an endpoint.
- If the indices are sorted topologically, indices increase from left to right along C.

⇒ Primal segments through c with larger slopes will have larger indices



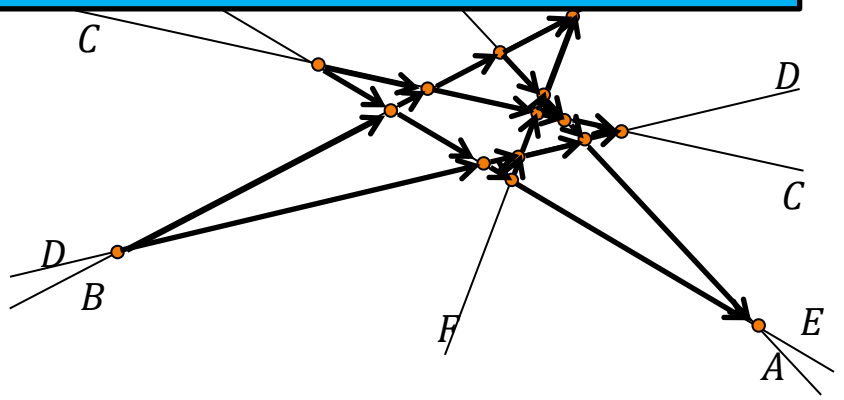
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}

Preprocessing:
Initialize the index and the set of vertices that can be assigned an index.



Visibility Graph [Welzl 1985]

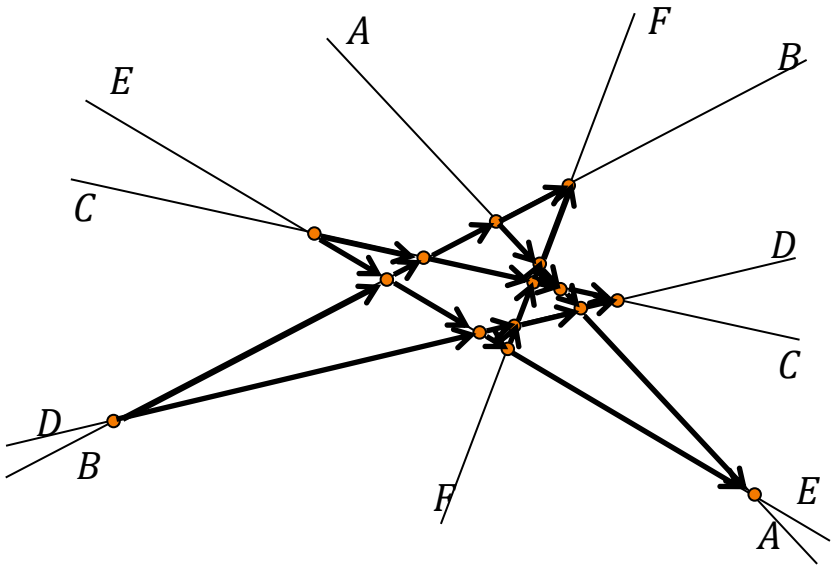
Topological Sort [Kahn 1962]:

Topo

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$

Until there are no vertices to assign:

- Assign the next index to the next vertex



Visibility Graph [Welzl 1985]

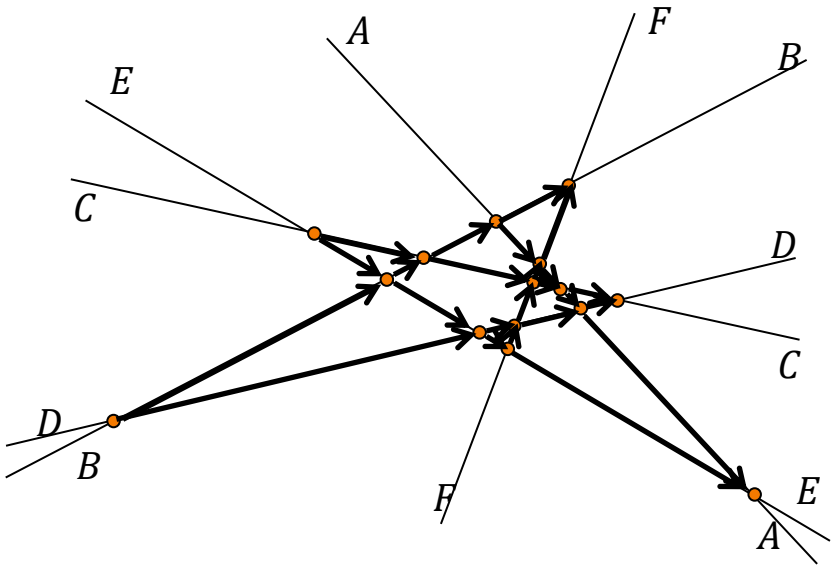
Topological Sort [Kahn 1962]:

Topo

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - $v \leftarrow$ Pop(S)
  - $i(v) \leftarrow c++$
  - $\forall\ (v,w) \in E$
    - $E \leftarrow E - \{(v,w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

Until there are no vertices to assign:

- Update the list of assignable vertices



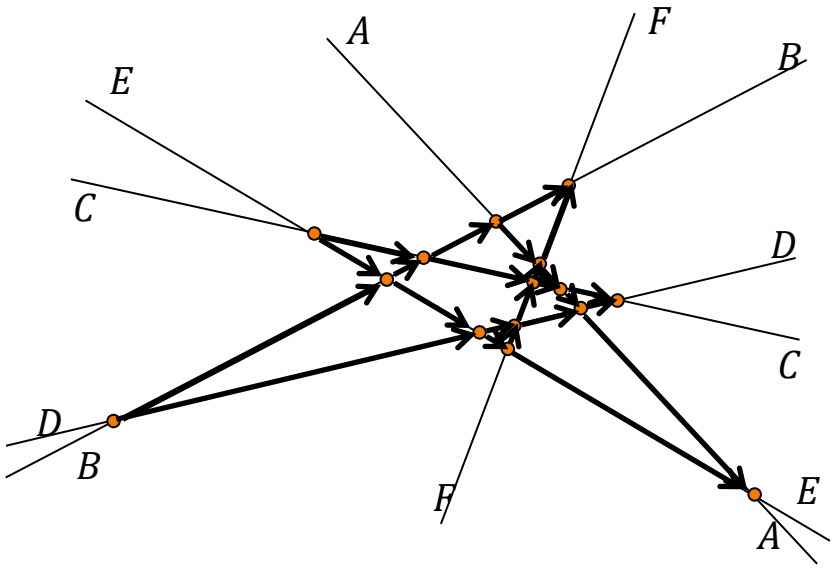
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

$S = \{BD, CE\}$

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$



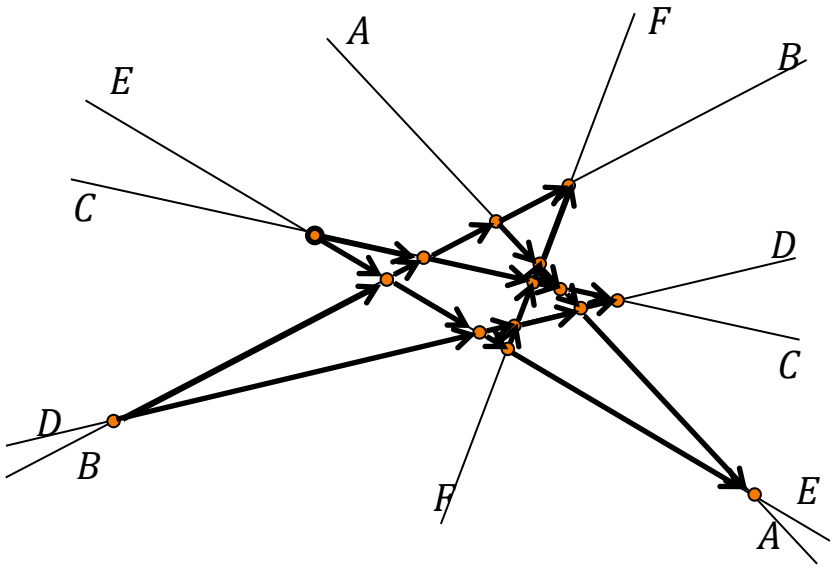
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{BD\}$
$v = CE$



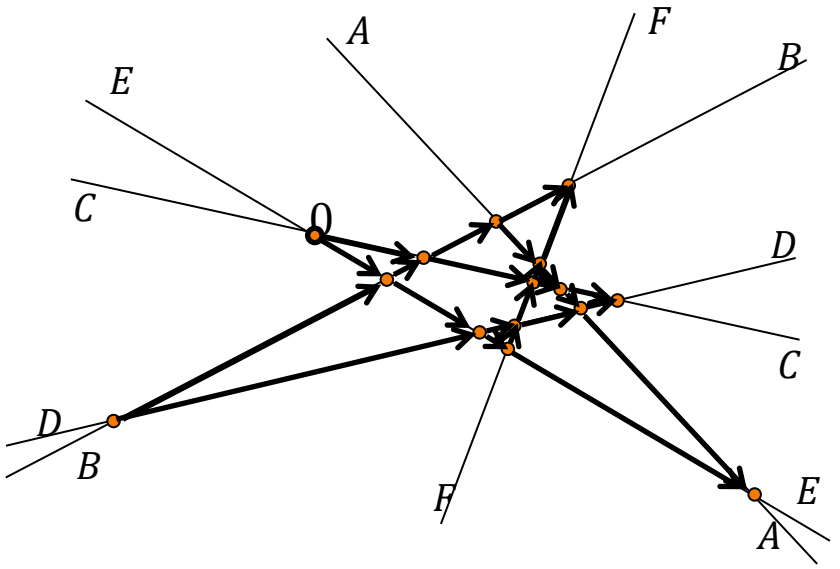
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{BD\}$
$v = CE$



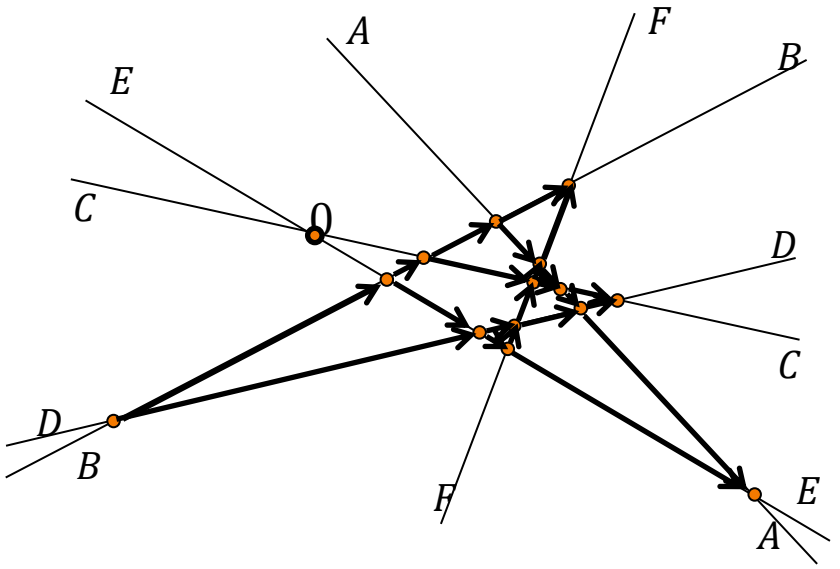
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $\boldsymbol{i(v) \leftarrow c++}$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{BD\}$
$v = CE$



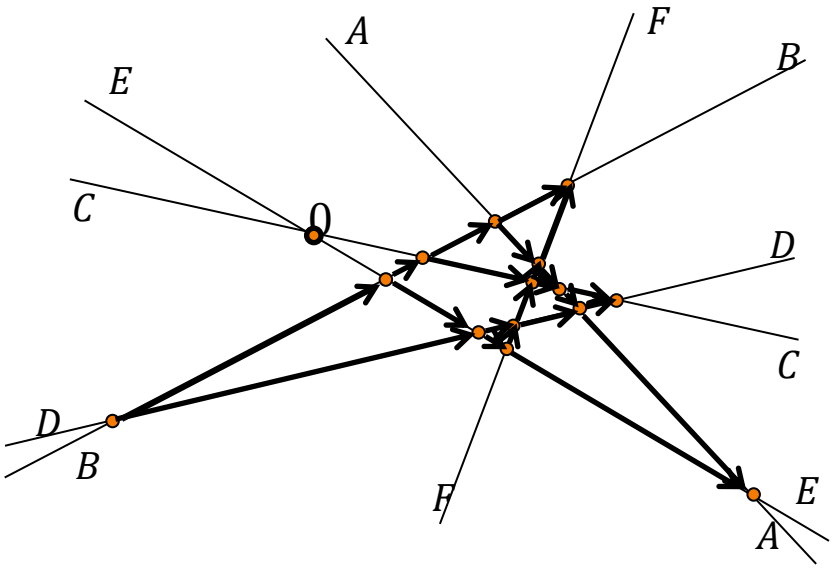
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $\boldsymbol{i(v) \leftarrow c++}$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{BD\}$
$v = CE$



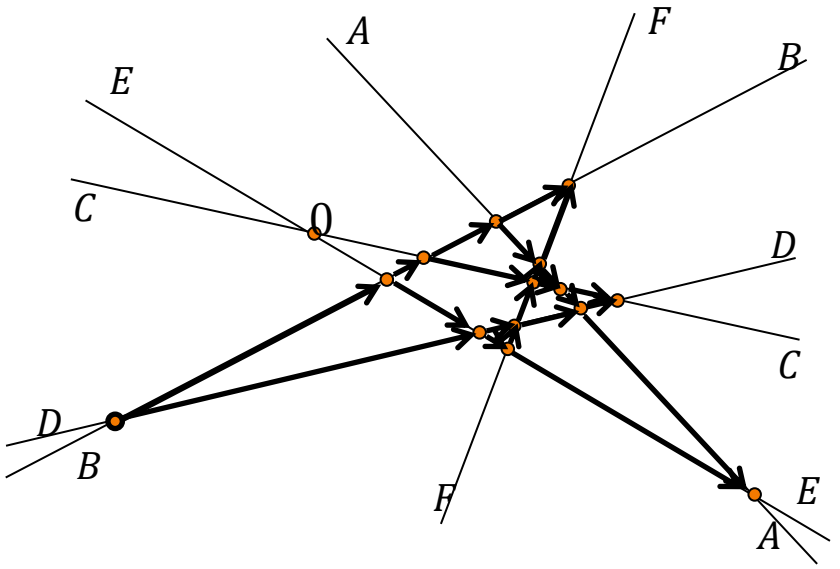
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{\}$
$v = BD$



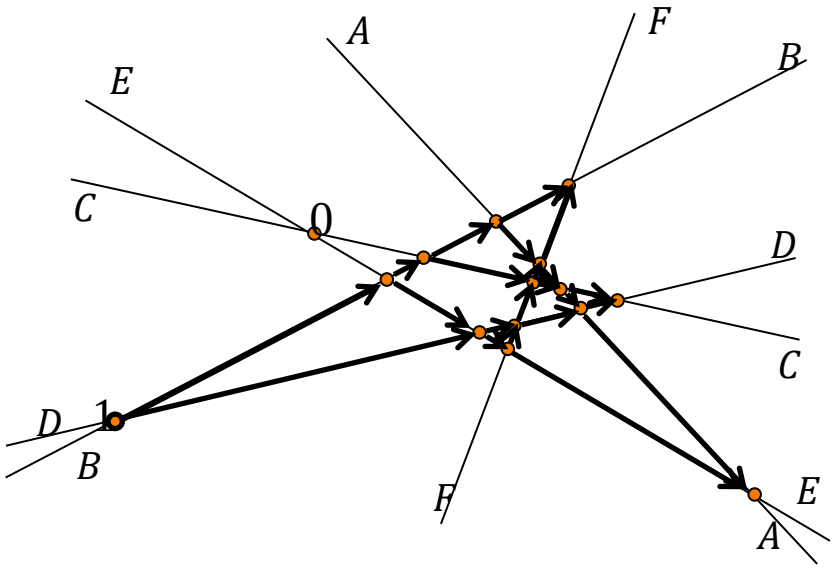
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{\}$
$v = BD$



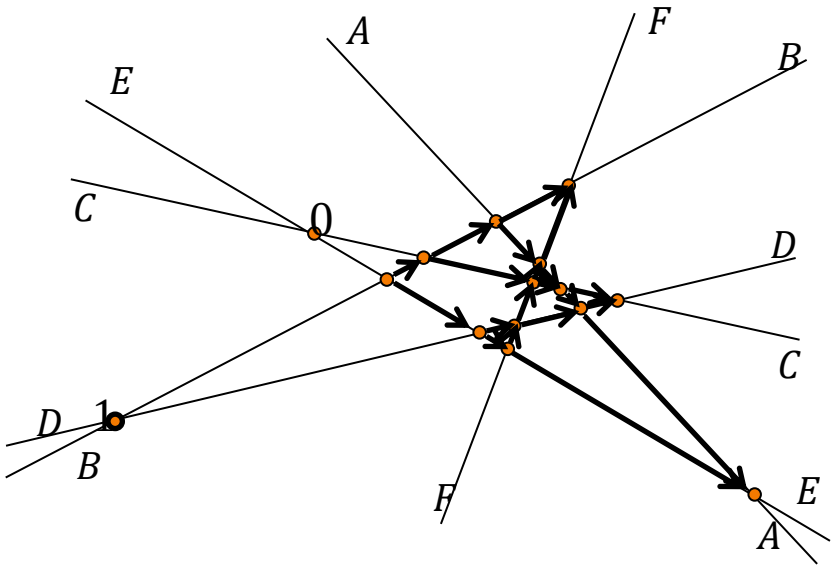
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $\boldsymbol{i(v) \leftarrow c++}$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{\}$
$v = BD$



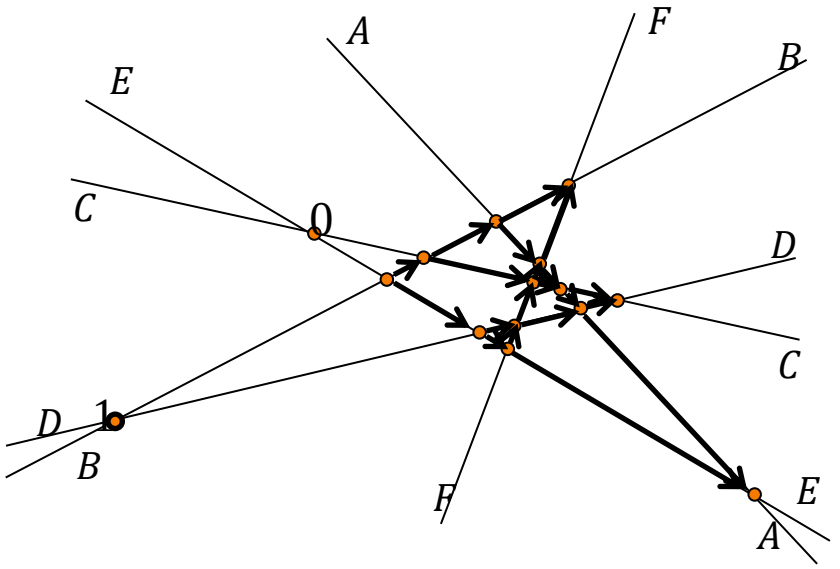
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{BE\}$
$v = BD$



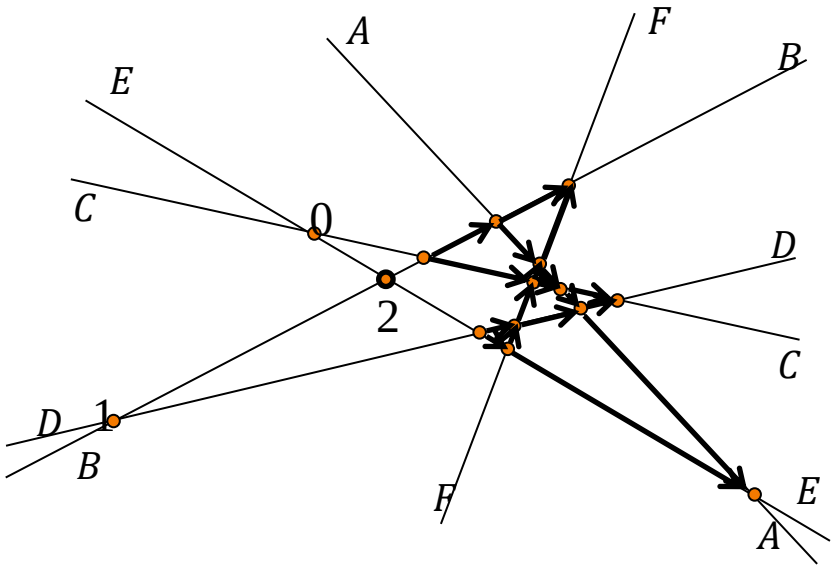
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{BC, DE\}$
$v = BE$



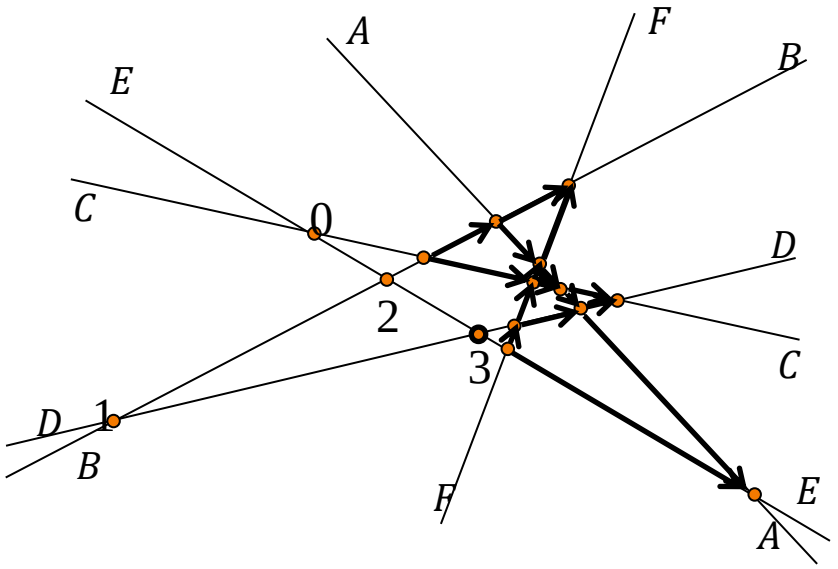
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{BC, EF\}$
$v = DE$



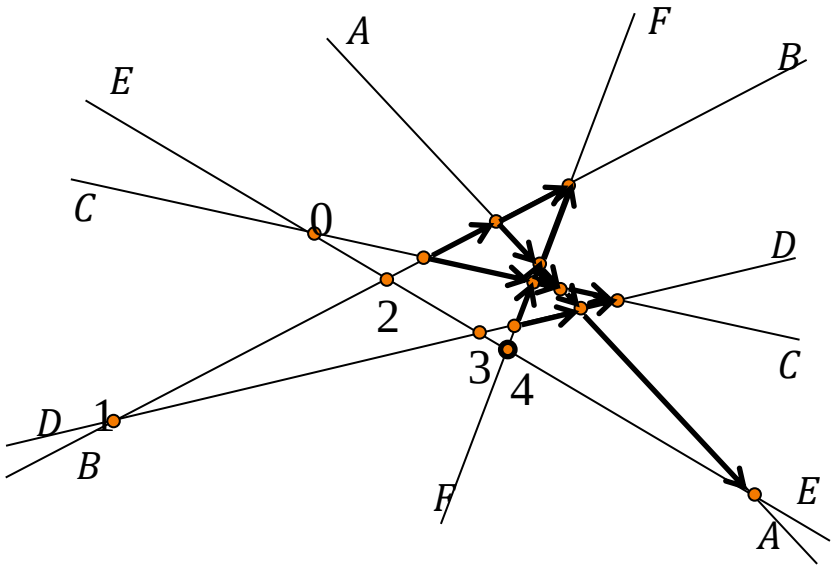
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{BC, DF\}$
$v = EF$



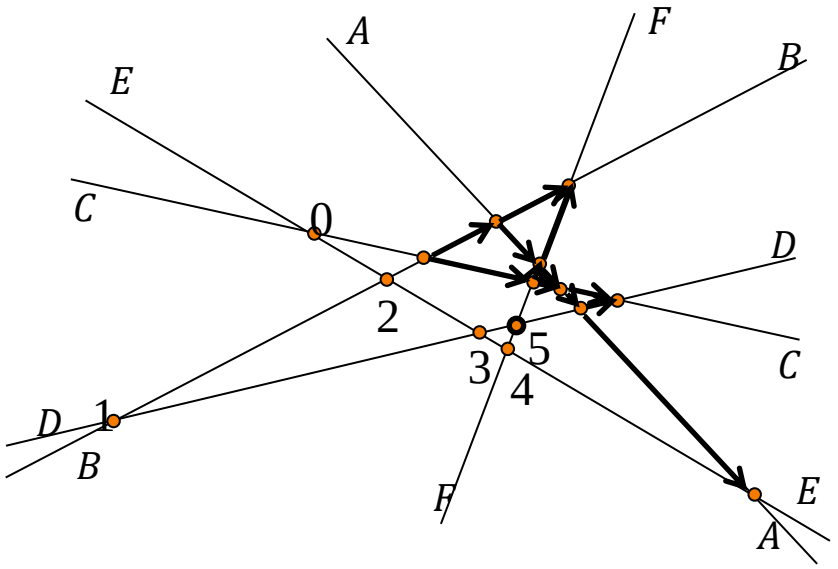
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{BC\}$
$v = DF$



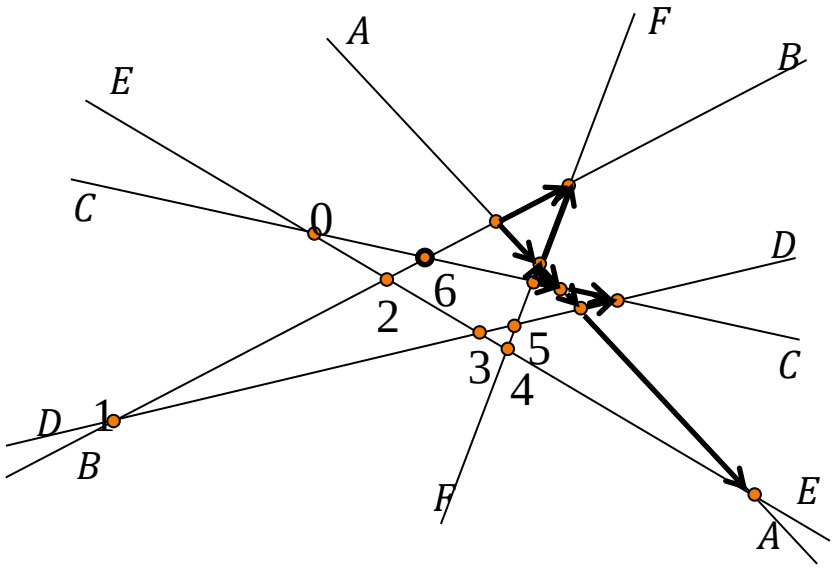
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{AB, CF\}$
$v = BC$



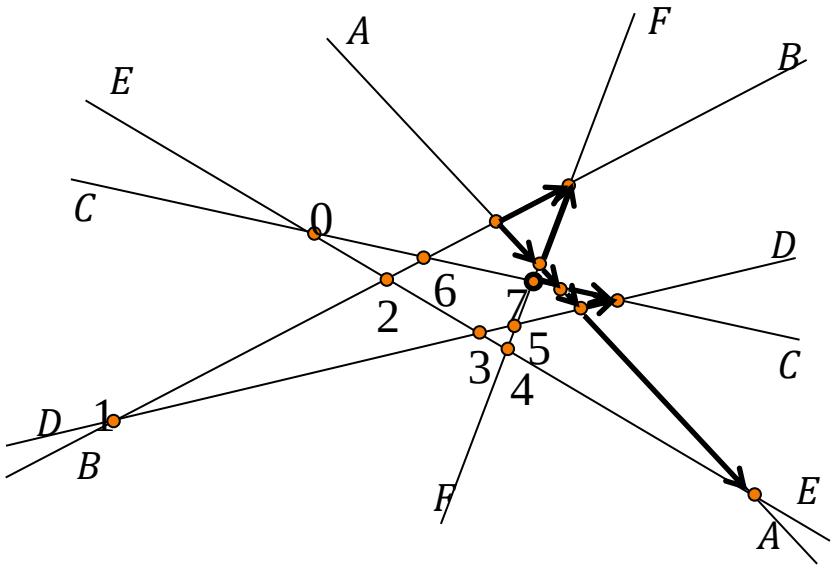
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{AB\}$
$v = CF$



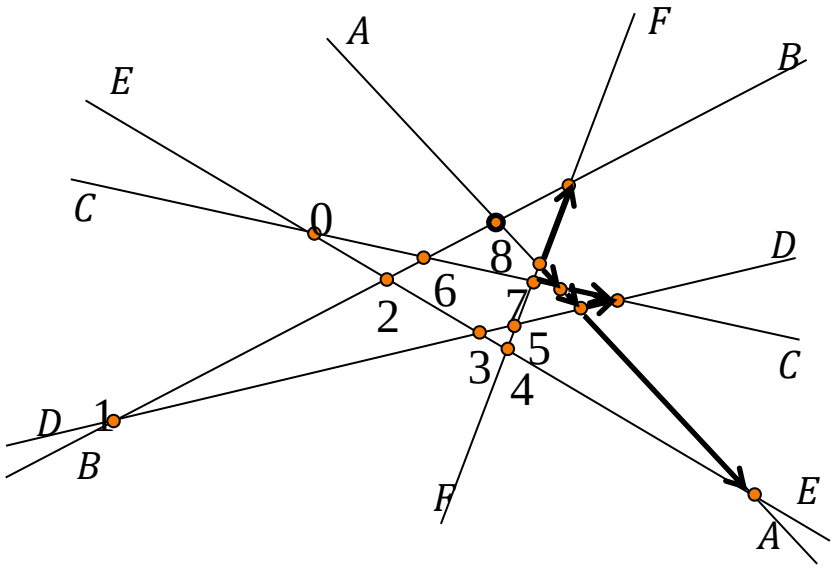
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{AF\}$
$v = AB$



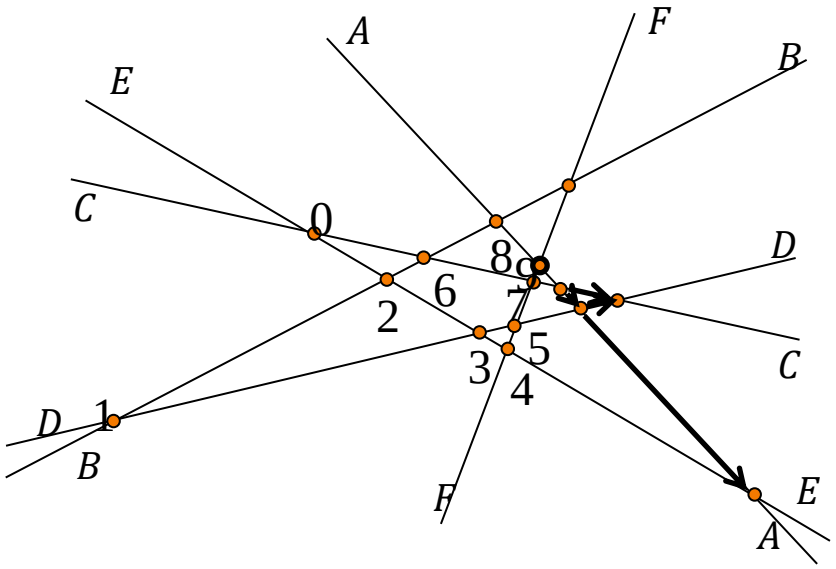
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{AC, BF\}$
$v = AF$



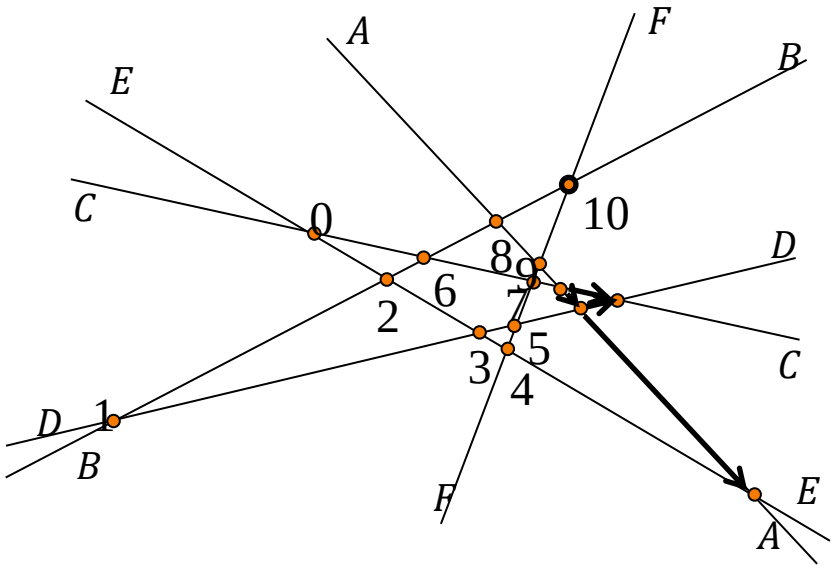
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{AC\}$
$v = BF$



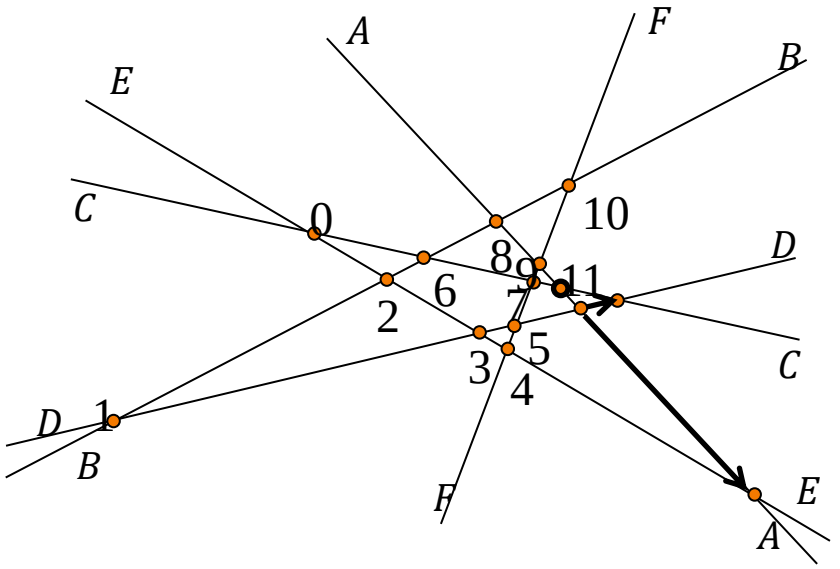
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{AD\}$
$v = AC$



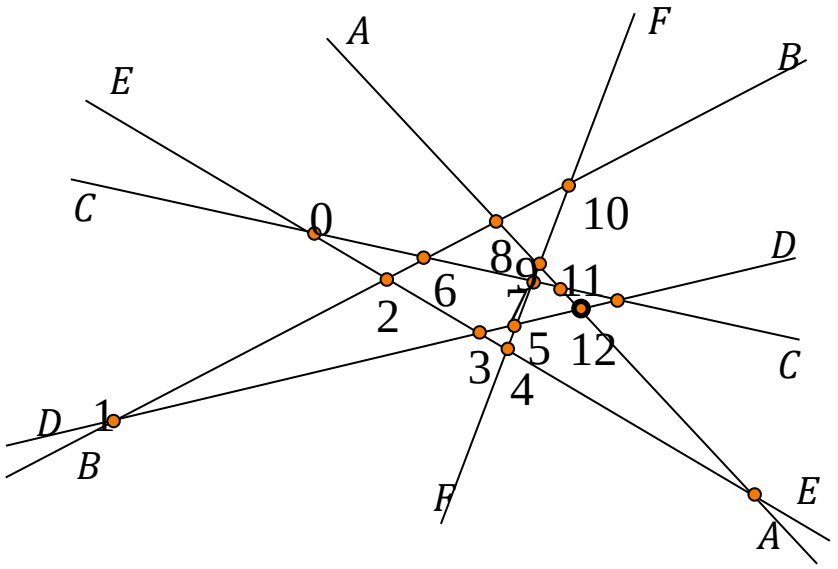
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{AE, CD\}$
$v = AD$



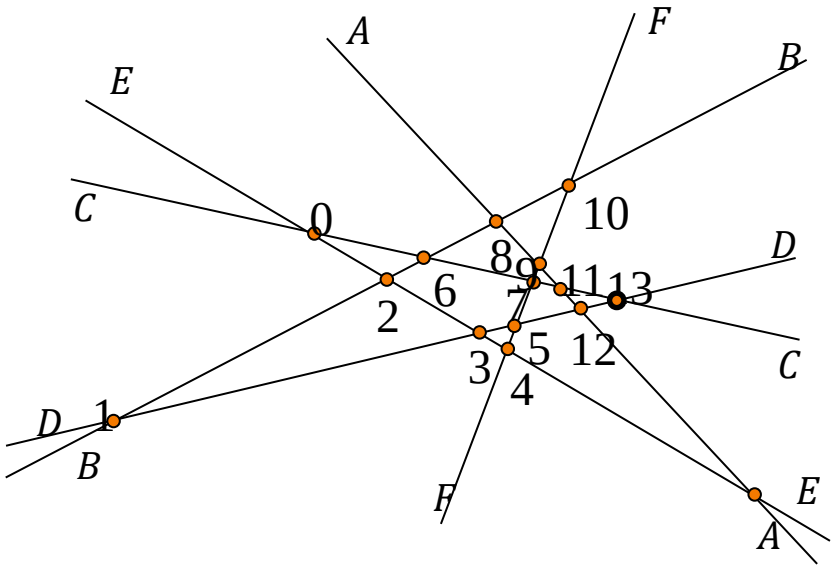
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{AE\}$
$v = CD$



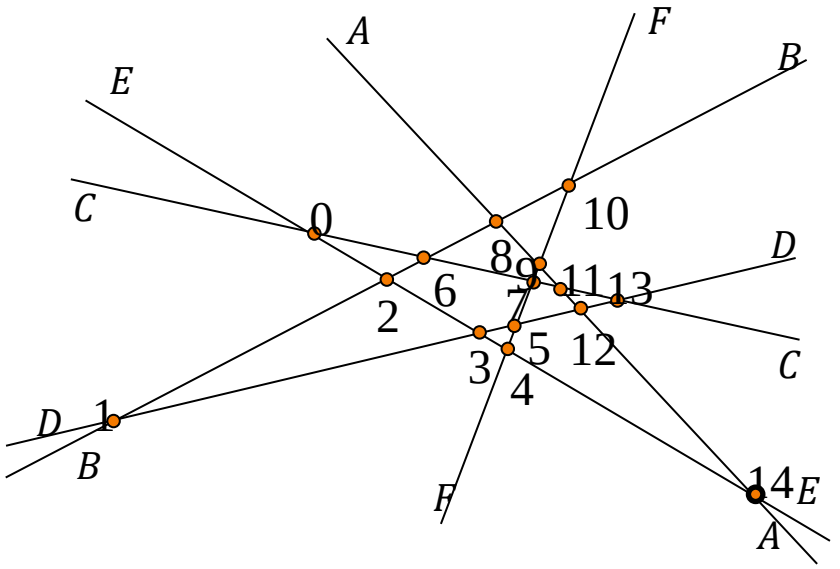
Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

$S = \{\}$
$v = AE$



Visibility Graph [Welzl 1985]

Topological Sort [Kahn 1962]:

TopoSort($G = (V, E)$):

- $c \leftarrow 0$
- $S \leftarrow$ {vertices with no incoming edges}
- while($S \neq \emptyset$)
  - » $v \leftarrow$ Pop(S)
  - » $i(v) \leftarrow c++$
  - » $\forall\ (v, w) \in E$
    - $E \leftarrow E - \{(v, w)\}$
    - if(Incoming(w)=$\emptyset$)
      - $S \leftarrow S \cup \{w\}$

Complexity: $O(|V| + |E|)$